2018

Enhancing Online Security with Image-based Captchas

Brian M. Powell

ENHANCING ONLINE SECURITY WITH IMAGE-BASED CAPTCHAS

Brian M. Powell

Dissertation submitted to the
Benjamin M. Statler College of Engineering and Mineral Resources
at West Virginia University

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Science

Afzel Noore, Ph.D., Chair
Edgar J. Fuller, Ph.D.
James D. Mooney, Ph.D.
George E. Trapp, Ph.D.
Frances L. Van Scoy, Ph.D.

Lane Department of Computer Science and Electrical Engineering

Morgantown, West Virginia
2018

Keywords: CAPTCHA, online security

ABSTRACT

Enhancing Online Security with Image-based CAPTCHAs

Brian M. Powell

Given the data loss, productivity, and financial risks posed by security breaches, there is a great need to protect online systems from automated attacks. Completely Automated Public Turing Tests to Tell Computers and Humans Apart, known as CAPTCHAs, are commonly used as one layer in providing online security. These tests are intended to be easily solvable by legitimate human users while being challenging for automated attackers to successfully complete. Traditionally, CAPTCHAs have asked users to perform tasks based on text recognition or categorization of discrete images to prove whether or not they are legitimate human users. Over time, the efficacy of these CAPTCHAs has been eroded by improved optical character recognition, image classification, and machine learning techniques that can accurately solve many CAPTCHAs at rates approaching those of humans. These CAPTCHAs can also be difficult to complete using the touch-based input methods found on widely used tablets and smartphones.

This research proposes the design of CAPTCHAs that address the shortcomings of existing implementations. These CAPTCHAs require users to perform different image-based tasks including face detection, face recognition, multimodal biometrics recognition, and object recognition to prove they are human. These are tasks that humans excel at but which remain difficult for computers to complete successfully. They can also be readily performed using click- or touch-based input methods, facilitating their use on both traditional computers and mobile devices.

Several strategies are utilized by the CAPTCHAs developed in this research to enable high human success rates while ensuring negligible automated attack success rates. One such technique, used by fgCAPTCHA, employs image quality metrics and face detection algorithms to calculate a fitness value representing the simulated performance of human users and automated attackers, respectively, at solving each generated CAPTCHA image. A genetic learning algorithm uses these fitness values to determine customized generation parameters for each CAPTCHA image. Other approaches, including gradient descent learning, artificial immune systems, and multi-stage performance-based filtering processes, are also proposed in this research to optimize the generated CAPTCHA images.

An extensive RESTful web service-based evaluation platform was developed to facilitate the testing and analysis of the CAPTCHAs developed in this research. Users recorded over 180,000 attempts at solving these CAPTCHAs using a variety of devices. The results show the designs created in this research offer high human success rates, up to 94.6% in the case of aiCAPTCHA, while ensuring resilience against automated attacks.

Dedicated to my family.

Acknowledgments

As I reach the end of my journey as a Ph.D. student, I would like to acknowledge those who have helped me along the way.

I am very thankful for the support of my advisor, Dr. Afzel Noore. He has been a constant source of encouragement and suggestions as I have conducted my research. His guidance and feedback has helped me to become a better researcher and computer scientist.

I would like to thank Dr. Richa Singh, Dr. Mayank Vatsa, Gaurav Goswami, and Adam Day for their collaboration on my CAPTCHA research. I also wish to acknowledge my committee members, Dr. Eddie Fuller, Dr. Jim Mooney, Dr. George Trapp, and Dr. Frances Van Scoy, for their support and suggestions. Finally, I wish to thank the faculty and staff of the Lane Department for their assistance.

The completion of my Ph.D. would not have been possible without the continued support of my family and friends. I thank them from the bottom of my heart.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

API	Application programming interface
CAPTCHA	Completely Automated Public Turing Test to Tell Computers and Humans Apart
COTS	Commercial off-the-shelf
DCG	Dynamic cognitive game
DoS	Denial of Service
GA	Genetic learning algorithm
HOG	Histogram of Oriented Gradients
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
OCR	Optical character recognition
ODBC	Open Database Connectivity
REST	Representational state transfer
SQL	Structured Query Language
SSIM	Structural Similarity model
UAC	User Access Control
URL	Universal Resource Locator
VIF	Visual Information Fidelity model

Chapter 1

Introduction

1.1 Motivation

Over the past 30 years, the Internet has grown from a small network used by the government and academic communities to become the backbone of global commerce and communications. This expansion has been rapid since the turn of the century: from an estimated 400 million people in 2000, the number of Internet users ballooned to 4.02 billion people by 2018 [1]. The value of online commerce is increasing at a similarly rapid rate. In 2015, retail e-commerce transactions alone totaled nearly $1.7 trillion and were expected to grow to $3.5 trillion by 2019 [2].

As high speed Internet connections have become widespread, individuals and businesses have leveraged the benefits of anywhere access afforded by the Internet by shifting more of their computing needs to the cloud and other online systems. Gartner, an information technology research firm, estimates that one-third of personal data, over one trillion gigabytes, is stored in the cloud [3]; 89% of enterprises store data on the public cloud and 17% of enterprises have large virtual machine deployments (over 1,000 virtual machines) running in public cloud infrastructures provided by companies such as Microsoft, Amazon, and Google [4, 5].

The shift to an increasingly online world is not without risk. Online and cloud computing systems are vulnerable to a wide array of threats and issues. Some, like the risk of data loss from physical failure or configuration problems, can affect any computer [6]. Others are specific to online systems or are easier for attackers to exploit because of remote access. Key concerns, summarized in Fig. 1.1, include:

1. **Uncontrolled access** can allow for inappropriate use of systems and wasted resources including bandwidth, processor time, and disk space [6, 7, 8]. Systems lacking proper access control can become host to spam posts, illegal file sharing, and in extreme cases, can even attack other systems.

2. **Denial of service (DoS) attacks** can prevent users from accessing systems and data [9]. There are several different types of attack, but most involve large amounts of traffic or requesting processor-intensive tasks that overwhelm networks and servers. This problem has even impacted systems at West Virginia University, where several services including the online directory were unavailable for extended periods due to DoS attacks [10].

3. **Compromised user accounts** can allow unauthorized access to systems and data. One common way of compromising accounts is by brute force or dictionary attacks where automated bots repetitively attempt to login to a system until they find a valid username and password combination [11]. Other attacks take usernames and passwords known from one system and attempt to use those credentials to login to other systems in case individuals reused their account information [12].

4. **Security vulnerabilities** can provide nefarious users with ways to access data, install software, change configuration settings, or crash systems [6]. New zero-day exploits, attacks that utilize previously unknown weaknesses to gain unauthorized access to systems, are now found on at least a monthly basis [13].

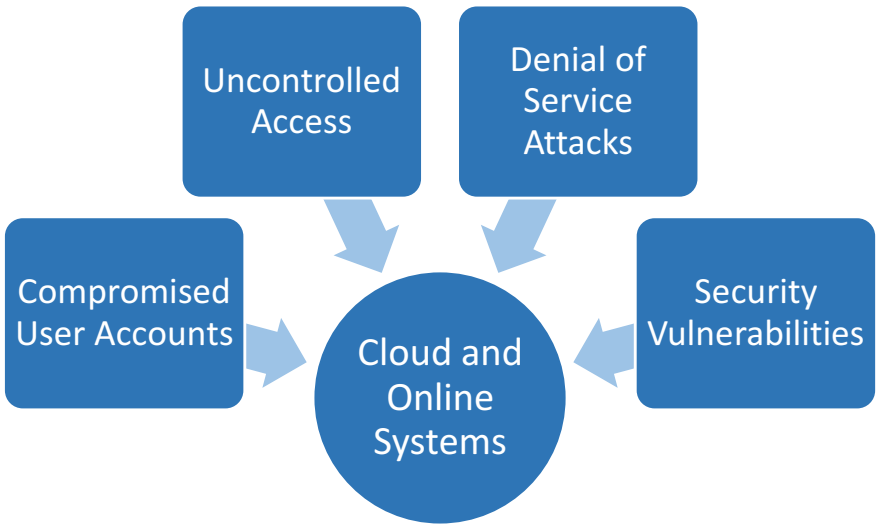


Figure 1.1: Major threats affecting online and cloud systems.

The costs associated with these threats are staggering. A 2016 study estimated the cost of a single data breach, such as what can occur with a compromised user account, at $4 million [14]. For midsize businesses, costs associated with denial of service attacks average about $40,000 per hour [15]. Even seemingly trivial issues like spam postings can result in intangible costs from damage to a company's reputation.

1.2 Goal

Given the economic and non-economic risks associated with data breaches, DoS attacks, and other security vulnerabilities, a great need exists for effective security tools to protect online systems from these risks. Tools that restrict access to protected systems to human users are particularly effective in ensuring system security. Many attack strategies rely on brute force; these attacks must be automated to be practical and cost-efficient. If automated access is prevented, many attacks become infeasible.

This dissertation seeks to develop effective security tools that prevent automated attacks without imposing more than a minimal impact on legitimate human users. Specifically, this dissertation proposes new forms of Completely Automated Public Turing Tests to Tell Com-

puters and Humans Apart (CAPTCHAs) [16] and the supporting tools necessary to facilitate their public use. CAPTCHAs are a type of human interactive proof that is implemented to restrict access to protected resources to users who are believed to be human. They require would-be users to complete tasks that are designed to be relatively easy for humans to complete while being challenging for automated software to perform correctly [17]. If users are able to successfully complete the required task, they are assumed to be human and likely legitimate, and, subject to any other applicable restrictions, are granted access to the protected resource.

As defined by Kluever [18], based on recommendations from researchers at Carnegie Mellon University [19], the Palo Alto Research Center [20], and Microsoft Research [21], effective CAPTCHAs must possess four properties:

1. **Automated:** It must be possible for a machine to automatically generate and grade the challenges.

2. **Open:** The database(s) and algorithm(s) used to generate the challenges must be publicly available to ensure that the difficulty of the CAPTCHA stems from the underlying hard artificial intelligence problem and not a secret algorithm.

3. **Usable:** Challenges should be easily and quickly solved by humans. The test should be as independent as possible of the user's language, physical location, educational background, and perceptual disabilities.

4. **Secure:** The underlying AI problem must be a well-known and well-studied problem where the best existing techniques are weaker than humans.

While CAPTCHAs have been in existence for over a decade, existing approaches have shortcomings relative to these essential properties, particularly in the areas of security and usability. These deficiencies have become severe as automated attack strategies have grown more sophisticated and as users move to mobile devices such as tablets and smartphones.

reCAPTCHA, one common existing approach, provides only a 75% human success rate while suffering from an automated attack success rate higher than 40% [22, 23]. Better solutions are clearly needed.

The CAPTCHAs proposed by this dissertation are designed to possess all four of Kluever's required properties so they can provide effective security for all users. These new CAPTCHAs provide human success rates of over 90% while having attack success rates approaching zero, all while being readily usable across both traditional and mobile computing platforms.

1.3 Contributions

This work provides the following set of original contributions:

1. **Design of effective CAPTCHAs facilitating easy use by legitimate human users on traditional and mobile computing devices while resisting automated attacks.** Existing CAPTCHA implementations, particularly those based on text-based tasks, have significant shortcomings in human usability, effectiveness as security tools, or both. These existing approaches are also difficult to complete with the touch-based input methods used on increasingly common tablets and smartphones [24]. The image-based CAPTCHAs proposed in this research overcome these challenges by utilizing tests based on face detection, face recognition, multimodal biometrics recognition, and object recognition. These tasks remain challenging for automated attackers to perform while leveraging skills that even newborn babies possess [25]. They are designed to allow easy use on desktop computers, laptops, tablets, and smartphones.

2. **Creation of a testing environment and associated tools for evaluating human performance on the CAPTCHAs proposed in this research.** A CAPTCHA Evaluation Platform consisting of a RESTful web service along with accompanying client-side tools, website, and supporting database were developed to facilitate one

of the largest human performance studies on CAPTCHAs completed to date. In total, more than 180,000 attempts were recorded through this platform as part of this research.

1.4 Organization

The remainder of this dissertation is organized as follows: Chapter 2 provides an overview of work relating to CAPTCHAs. Chapters 3-8 describe specific contributions in CAPTCHA design. Chapter 9 details the evaluation platform developed to facilitate testing and analysis of the CAPTCHAs created in this research. Finally, Chapter 10 concludes the work by providing a summary of accomplishments as well as future directions for research.

Chapter 2

Background

This chapter reviews the history and development of existing CAPTCHAs and online security techniques. Its intent is to provide readers with a basic understanding of relevant scholarship in this area.

2.1 Non-CAPTCHA online security techniques

While a number of solutions exist to counter threats facing online computing systems, many are not adequately scalable or sufficiently sophisticated to prevent attacks. Let us consider several existing techniques.

2.1.1 Restricting anonymous access

One common strategy for protecting online systems is to require users to sign-in with a user account to perform certain restricted actions. The actions that require a login vary by system. For example, a newspaper website may allow all visitors to read articles but require they login to post comments on a story to avoid spam postings. Requiring logins can help protect against abuse of resource-intensive tasks, such as complex database queries, that can be used to cause denial of service [9].

This solution comes with a cost. Many users find account creation and login requirements burdensome and will avoid using these systems [26]. There is additional overhead involved in maintaining user accounts and they do not entirely solve the problem. Spammers and attackers have developed means of automating the account creation process, allowing them to create new logins as needed [27]. In short, legitimate users are the only individuals being impacted by account requirements.

2.1.2 Protecting account login systems

For systems requiring logins, one common approach for preventing brute force attacks is to lock out accounts that have reached a specified number of unsuccessful login attempts [28]. When an account is locked out, it cannot be used to login. While this approach is beneficial in slowing or preventing brute force-based attacks, an undesirable side effect is that it can be used as a Denial of Service attack (DoS) attack [29]. Attackers can force a lockout of the accounts of legitimate users, thereby preventing them from accessing the system.

Another approach is to implement multifactor authentication. With this technique, users must perform an additional verification task to prove who they are beyond entering a username and password. For example, they may be asked to type-in a value from a SMS text message [30] or from an app on their mobile device [31]. Attackers should not have access to these verification codes, and as such, will be unable to login to the protected system. Since the verification step generally occurs after the username and password have been verified, would-be attackers would know they have a valid username and password they could then attempt to use on other systems. Since many users reuse credentials across systems, attackers may be able to login to other systems not protected by multifactor authentication [32].

2.1.3 Protecting against security vulnerabilities

While the abilities that an attacker gains upon exploiting a software vulnerability vary depending on the issue they exploit, it is safe to assume they have at least the same ability to use system resources as the user account under which the compromised software runs. For this reason, a good security practice is to use the principal of least privilege in granting account permissions; accounts should receive the bare minimum permissions required to complete a task [33].

Since it is inconvenient to constantly change user accounts, however, almost all users run under accounts with more permissions than required [33]. Some services must run under highly privileged accounts to fulfill their duties, which makes them a particularly large risk if they can be compromised. Microsoft attempted to mitigate these risks in Windows with User Access Control (UAC), a prompt that requires users to grant permission for potentially dangerous operations such as installing software and modifying system files [34]. When the system is being used locally, this is effective since Windows will only accept input for the UAC prompt from the keyboard or mouse, not from other software like a compromised application. When accessed remotely via services such as Remote Desktop or Citrix, UAC is vulnerable to attack since an attacking bot could spoof the remote user's input.

2.2 CAPTCHAs

CAPTCHA, Completely Automated Public Turing Tests to Tell Computers and Humans Apart, are a type of human interactive proof employed to prevent automated access to online systems [35]. They represent one solution to the limitations and side effects of the security techniques discussed in Section 2.1. To gain access to protected systems, CAPTCHA users must successfully complete tests designed to be readily solvable by human users but difficult for computers to perform. The difficulty that computers encounter in solving CAPTCHAs is what makes these tests an effective security tool. Many threats to online systems, including

those discussed in Section 1.1, rely on automated attacks to be feasible. An attacker who must manually enter each password guess is not going to be able to test many passwords, and as a result, will unlikely ever compromise a user account. A spammer who has to manually send each spam message is not going to be able to send much spam. CAPTCHAs eliminate the ability to have automated bots perform these tasks, and without the bots, the attacks become too costly to be worthwhile.

2.2.1 Turing tests

As their name indicates, CAPTCHAs are based on Alan Turing's seminal 1950 proposal of what has come to be known as the Turing test. In [36], Turing proposes an "imitation game" where a human questioner interacts with two other participants via text-based communication. One participant is human, the other a computer. If the computer is able to provide responses that are sufficiently advanced to the point that the human questioner is unable to determine whether the participant is human or a computer, the computer passes the Turing test.

CAPTCHAs employ a variation on this problem, sometimes referred to as a reverse Turing test [37]. Instead of the questioner being a human as in the original Turing test, with CAPTCHAs, it is a computer that generates and evaluates participant responses to the tests. The CAPTCHA system asks participants to complete tests based on computationally challenging tasks that are only expected to be solvable by humans [19]. If they are able to complete the test, the CAPTCHA system assumes the participants are human and grants access to the protected resource.

CAPTCHAs can be evaluated on two primary performance metrics, human success rates and automated attack success rates. In designing CAPTCHAs, the goal is to maximize the human success rate while minimizing the automated attack success rate. To be effective, a CAPTCHA's automated attack success rate must be below 1% [38]. Human success rates of at least 80% are preferred [39].

2.2.2 Types of CAPTCHAs

Existing CAPTCHA implementations can be divided into one of seven categories [40, 41, 42]:

1. **Text-based CAPTCHAs** present users with text that has been visually distorted or deformed. To solve the CAPTCHA test, users must accurately perform character recognition and enter the text that is displayed [16].

2. **Image-based CAPTCHAs** can take several forms, but most are based on image categorization tasks [18]. They generally require users to either identify embedded images matching a specified descriptor (e.g., selecting cats or dogs [43]) or to categorize existing unlabeled images.

3. **Video-based CAPTCHAs**, like image-based CAPTCHAs, are generally based on categorization and labeling tasks. Users are instructed to watch a video and then select an appropriate category or tag describing the video [18].

4. **Audio-based CAPTCHAs** most commonly require users to identify and type-in letters or numbers that are read aloud in an audio file. They are frequently used as an alternative to other CAPTCHA types for users with visual impairments [44].

5. **Interactive CAPTCHAs** require users to directly interact with and manipulate elements of the CAPTCHA, most commonly by dragging and dropping items. This category also includes CAPTCHAs that present the CAPTCHA task as a game [45].

6. **Problem-based CAPTCHAs** can take many forms including linguistic problems that ask users to fill-in missing words in sentences, make sense of text, or answer simple common knowledge-type questions [41]. Others may ask users to solve mathematical problems or to utilize semantic relationships to organize objects [46, 47].

7. **Behavior analysis-based CAPTCHAs** analyze the system environment and user behavior, sometimes including factors outside of the CAPTCHA test itself, to deter-

mine if the user is human or computer [48]. They frequently incorporate other types of secondary CAPTCHA tests.

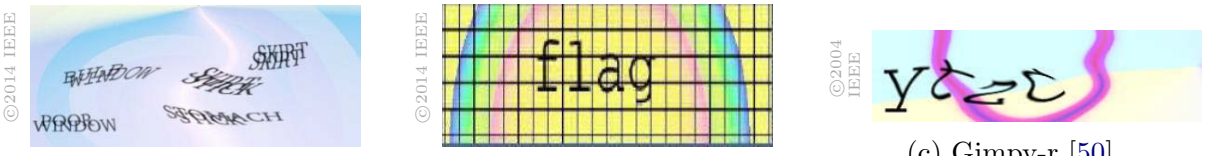
2.2.3 Text-based CAPTCHAs

Text-based CAPTCHAs represent the most common implementation. These CAPTCHAs graphically render a string of text designed to be difficult to evaluate using optical character recognition (OCR) algorithms. The distorted text is shown to the user as an image file. The user must correctly type-in the text to solve the CAPTCHA [16].

Early CAPTCHAs

The very first CAPTCHAs were created in response to automated bot-driven "spam" submissions that negatively impacted online services offered by AltaVista and Yahoo! [18, 20]. The *AltaVista CAPTCHA* was designed by reverse engineering OCR technology to focus on its weaknesses [18]. The CAPTCHA generates a ransom note-style image consisting of several alphanumeric characters, each rendered in a different font, and placed in random positions on a colored stochastically generated background [49]. AltaVista attributed an "over 95%" drop in spam submissions to the CAPTCHA at the time of its deployment [20].

Researchers at Carnegie Mellon University coined the term CAPTCHA when designing a solution to Yahoo!'s problems with automated bots [20]. Their solution is the Gimpy family of CAPTCHAs shown in Fig. 2.1. *Gimpy* renders seven visually squished, stretched, or skewed English words on a colorful background. *EZ-Gimpy* renders single English words and the *Gimpy-r* variant uses a random string of four characters [50]. With attacks against these CAPTCHAs having up to a 99% success rate [50], none of these approaches are secure. Success of these attacks can be traced to two major weaknesses: (1) use of dictionary words in the CAPTCHA, allowing an attacker to narrow its guesses [51], and (2) easily segmentable characters, allowing individual letters to be identified by comparing shape outlines [50, 52].



(a) Gimpy [53] (b) EZ-Gimpy [53] (c) Gimpy-r [50]

Figure 2.1: Samples of the Gimpy family of CAPTCHAs.

Character degradation CAPTCHAs

Character degradation CAPTCHAs add noise to CAPTCHA images so character outlines become indistinct. This helps defeat OCR and character matching-based attack strategies. *Pessimal Print* [37] adds salt-and-pepper background noise and random variation in character segment widths to simulate the changes that can occur with multiple generations of photocopying a document. In testing by its developers, humans solved 100% of Pessimal Print CAPTCHAs. However, the CAPTCHA also has a high success rate for attackers. By leveraging Pessimal Print's use of dictionary words, the Mori-Malik attack [51] could defeat the CAPTCHA with a 40% success rate [54].

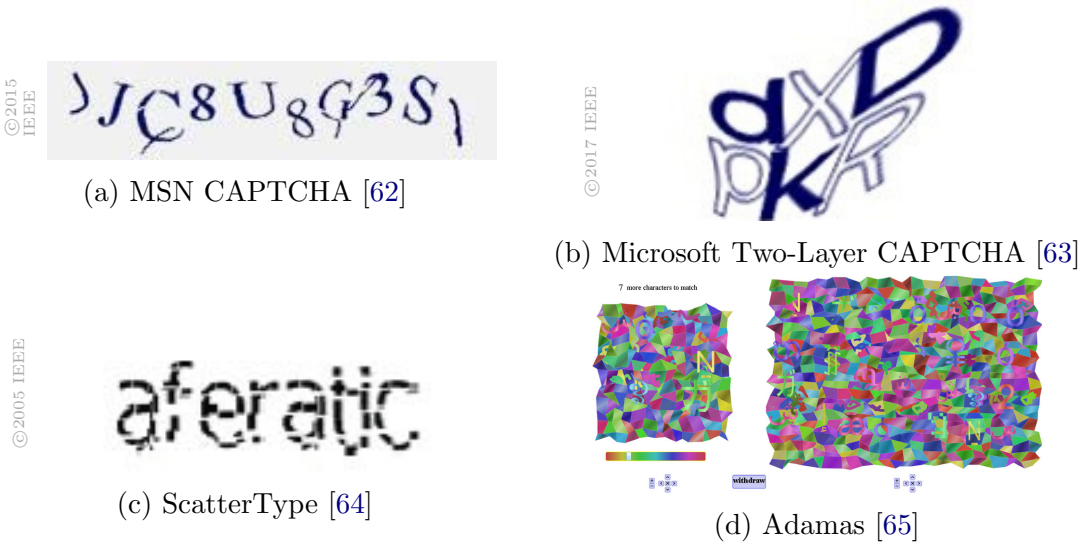
BaffleText [54] overlays a black-and-white image of geometric shapes such as squares, circles, and ellipses on top of a text string. A difference operation is conducted where the text and geometric shapes intersect, yielding sections of white text on a black background and black text on a white background. While this approach was not successful in stopping automated attacks on its own with a 25% attack success rate [54, 55], its high 89% human success rate has led to similar difference operations being incorporated into other CAPTCHAs as an added distortion [56].

The open source *Securimage* CAPTCHA [57] creates monochrome images containing either two words or a randomly generated string. The characters are warped and distorted, and then speckle noise and random lines are added to the entire image. In testing, human users solved 99% of CAPTCHAs based on English dictionary words but only 74% of those based on random characters [58]. Securimage's easily segmentable text is highly susceptible

to neural network-based attacks on its letterforms, which achieve a near 100% attack success rate [59]. Its English word variant is also vulnerable to dictionary-based attacks.

Segmentation-resistant CAPTCHAs

When the letters in a text-based CAPTCHA can be individually recognized, the task of breaking the CAPTCHA devolves to simple single character recognition with accuracy rates approaching 100% [60, 61]. Segmentation-resistant CAPTCHAs include features designed to make isolation of individual characters challenging. Would-be attackers are required to first segment the CAPTCHA before they can attack the individual character forms, which is a more difficult task. Several examples of these CAPTCHAs are shown in Fig. 2.2.



(a) MSN CAPTCHA [62]

(b) Microsoft Two-Layer CAPTCHA [63]

(c) ScatterType [64]

(d) Adamas [65]

Figure 2.2: Examples of segmentation-resistant CAPTCHAs.

MSN CAPTCHA [66] represents one attempt at leveraging the segmentation problem. It combines traditional CAPTCHA techniques such as character warping, rotation, and varying the stroke thickness with changing the positions of the letters and adding arcs running through multiple letters. Unfortunately, significant predictability in the text length and positioning in generated CAPTCHA images enabled the creation of a process to methodically segment, deconstruct, and recognize the CAPTCHA with over 90% accuracy [60]. A later

variant of this CAPTCHA, the *Microsoft Two-Layer CAPTCHA* [63] attempts to use warping effects, hollow characters, and close placement of its letters to prevent segmentation. It is not much more secure, with an attack based on a series of systematic image processing operations and use of a neural network achieving a 28% attack success rate. The *Megaupload CAPTCHA* [67], which rotates and overlaps individual characters in an attempt to avoid segmentation, has also proven readily attackable with 78% accuracy. Its treatment of coloring overlapping areas of text in white and non-overlapping areas in black inadvertently provides outlines for attackers to use in the segmentation process.

ScatterType [68] takes the opposite approach of other segmentation-resistant CAPTCHAs. Rather than connecting multiple characters, it divides six to eight individual letters into multiple pieces which are then visually scattered. Users must be able to reassemble the pieces to form letters to solve the CAPTCHA. While this approach has human success rates approaching 95%, three principal means of attacking this CAPTCHA have been identified [64]. The attack vectors include reassembling the character fragments as in a jigsaw puzzle, leveraging that the generated text strings are not fully random, and performing outline recognition based on the use of one typeface (from a limited pool) to render the original words.

The commercial *BotDetect CAPTCHA* randomly applies one of 60 different distortion techniques to its generated tests [69]. Its approaches include overlapping characters, adding connecting lines, distorting or dividing letterforms, and varying character colors [70]. Despite this wide variety of potential distortions, this CAPTCHA has been attacked using multiple strategies including a computer vision-based approach achieving a 64% success rate [69, 71].

Several CAPTCHAs attempt to prevent segmentation by embedding text in complicated backgrounds. *Anti-SIFT CAPTCHA* [72] embeds multicolored text into photographic backgrounds, while *Adamas* [65] generates a geometrically complex multicolored background with randomly shaped pieces. Six to eight multicolored distorted Unicode characters are placed onto this background. The user is presented with the test image and a virtual keyboard containing 25-30 Unicode characters including the test image's characters and visually similar

Unicode homoglyphs, presented on the same style of background as the test image. Users are directed to match the characters on the virtual keyboard to those in the test image by clicking on the matching pairs. Humans achieved a 77.5% success rate in solving Adamas, but they took an average of 111 seconds to complete an attempt. This exceptionally long time is likely to drive away legitimate users and outweighs the CAPTCHA's 0% automated attack success rate.

3-D CAPTCHAs

The *3D* [73], *tEABAG_3D* [74], *Ku6* [75], and *Tencent* [75] CAPTCHAs attempt to prevent optical character recognition by rendering text as a three-dimensional object that can be visually transformed or embedded into a background. These approaches are partially successful at resisting simple OCR-based attacks, but Nguyen et al. [76] have developed a novel attack which systematically removes the three-dimensional elements and converts the CAPTCHA image to a simple text string that can be attacked with success rates of up to 76% for tEABAG_3D. Ye et al. [75] have achieved a 51% attack success rate against Ku6 CAPTCHA and a 69% rate against the Tencent CAPTCHA. *Enhanced STE3D-CAP* [77] extends the 3D CAPTCHA idea by rendering the text as a stereoscopic image. Special 3D stereoscopic glasses are required to view the CAPTCHA, limiting its practicality.

Real world imagery-based CAPTCHAs

While most text-based CAPTCHAs attempt to artificially construct text unrecognizable by OCR technology, some CAPTCHAs instead base their tests on real world images of text. *Handwritten CAPTCHA* [78] relies on a database of English handwritten city names and character images taken from U.S. mail packages not automatically identified by the U.S. Postal Service's mail sorting system. Individual characters are combined to create words, and both the natural and artificially constructed words are distorted. Humans correctly identified over 82% of these images, while automated attacks succeeded up to 12% of the time. An

extension of this work, *Synthetic Handwritten CAPTCHA* [79], uses artificially generated cursive letters. While the synthetic handwriting had a lower 3% automated attack rate, the human success rate was sharply lower compared to Handwritten CAPTCHA. *Multilingual Handwritten CAPTCHA* [80] attempts to address the language dependency issues present in the earlier handwriting CAPTCHAs by presenting users with CAPTCHAs from one of four languages (English, Spanish, French, or Arabic) as appropriate for their detected location.

The original version of *reCAPTCHA* [81], shown in Fig. 2.3, takes a similar approach to Handwritten CAPTCHA by using images of text from book digitization projects and street addresses from Google Maps Street View photographs that could not be identified by OCR technology [82]. Google uses the results collected from human users to perform useful work; the previously unknown values are tagged with the human-provided responses using a consensus-driven process. reCAPTCHA has become one of the most popular CAPTCHAs [83]. As a result, it has attracted significant attention in breaking the CAPTCHA both from the academic community [56, 23] and would-be attackers. Attack strategies have been developed with an over 40% success rate [23]. Attempts to counter attackers with increased distortions have led to complaints that reCAPTCHA is becoming too challenging for humans to solve [84].

Figure 2.3: Sample reCAPTCHA test. [85]

2.2.4 Image-based CAPTCHAs

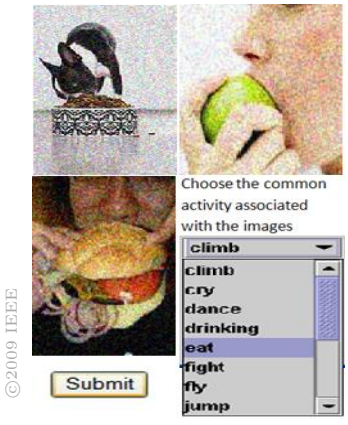
As optical character recognition technology has improved, text-based CAPTCHAs have become locked into escalating the intensity of distortions applied to the characters. This esca-

lation negatively impacts human performance. To avoid this issue, there has been interest in developing CAPTCHAs relying on a different mode of operation. One such approach is to require users to identify or understand objects depicted in images.

Discrete image classification CAPTCHAs

The most common type of image-based CAPTCHA requires users to classify objects present in discrete images. These CAPTCHAs generally follow one of two approaches: (1) users identify a category for images they are shown or (2) users select from an image or set of images those which match a specified description. *ESP-PIX* [53] and *Activity Recognition CAPTCHA* [86], illustrated in Fig. 2.4, are examples of the first approach. They present users with a small set of images to categorize using predefined options from a dropdown list. While this approach is simple, it is also highly vulnerable to attack. Even a random guess by an attacker has a non-trivial chance of being correct. *Naming CAPTCHA* [87] attempts to address this shortcoming by providing users with a freeform text box to type-in a descriptor for the images shown. Even after adjusting for spelling errors and synonyms, this CAPTCHA achieved only a 74% human success rate in limited testing.

(a) ESP-PIX [53]



(b) Activity Recognition CAPTCHA [86]

Figure 2.4: Samples of discrete image classification CAPTCHAs where users choose an appropriate category to describe presented images.

KittenAuth [88], *Asirra* [43], *SQUIGL-PIX* [89], *Simple Visual CAPTCHA* [90], *Confident CAPTCHA* [91], and *Relacha* [92] are examples of the second approach. Users are instructed to select from a set of distinct images those which match a specified criteria, such as kittens in KittenAuth or dogs and cats in Asirra. The presentation of distinct standalone images, rather than a composite image, makes these approaches subject to straightforward attacks using image classifiers. One image classifier-based approach has an 82.7% accuracy rate in discriminating between dogs and cats as used by Asirra [93].

EmojiTCHA [94] asks users to categorize individual images rather than groups. It presents users with a visually distorted photograph of a person's face. The user is then asked to select from a predefined list the emoji symbol which best represents the emotion shown by the person in the photograph (e.g., happiness, sadness, surprise, anger). The CAPTCHA's authors suggest using a series of EmojiTCHA tests to reduce the likelihood of brute force guessing, but even when using three tests with three emotions each as they suggest, the likelihood of random guesses solving the CAPTCHA is still a significant 3.7%.

Composite image classification CAPTCHAs

Several image classification CAPTCHAs rely on composite images to reduce the likelihood of image classifier-based attacks. *CaptchAll* [95] presents users with an image containing multiple embedded objects. Users are prompted to click on specific items within the composite image to solve the CAPTCHA. Its authors do not provide human usability data, but note a brute force attack success rate of less than 0.03%. *Scene Tagging CAPTCHA* [96] implements a more advanced approach incorporating three different types of tests. The first asks users to click on a specified object (e.g., select a soccer ball). The second asks users to identify an object by its relationship to another object (e.g., "name the object that is directly to the upper-left of the butterfly"). The third asks the user to specify how many objects of a given type are present. Both of the latter types of questions provide users with a list of predefined answers, and, consequently, suffer from the same high brute force guess

rate as ESP-PIX. The complexity of Scene Tagging CAPTCHA's operation also increases difficulty for human users in correctly understanding and solving the test.

Figure 2.5: Sample of IMAGINATION CAPTCHA [97].

The *IMAGINATION* CAPTCHA [97], shown in Fig. 2.5, uses a two-step process where users select the geometric center of an image embedded in a larger composite image and then label the selected image used a predefined list. The CAPTCHA has a 4.95% automated attack success rate [98], while providing only a 70% human success rate [97] due to the CAPTCHA's complexity. In testing conducted for this research, users indicated the radio buttons and long list of predefined options made IMAGINATION cumbersome to complete on a mobile device. *Implicit CAPTCHA* [99] is also challenging to use on mobile devices. This CAPTCHA presents users with a large image from which they are directed to select a specific small area, which can be difficult to complete on smartphones with small touchscreens [100].

Face-based CAPTCHAs

Face detection and face recognition have long been challenging tasks for computers, which makes them natural tests to use in CAPTCHAs. In [101], Misra and Gaj propose a face recognition CAPTCHA that presents users with six discrete images. Three people are shown

twice, once in the first column and once in the second column, as depicted in Fig. 2.6. Users match the images of the same people to solve the CAPTCHA. Because of its design, the CAPTCHA has a high 1-in-6 random guess success rate that renders it insecure.

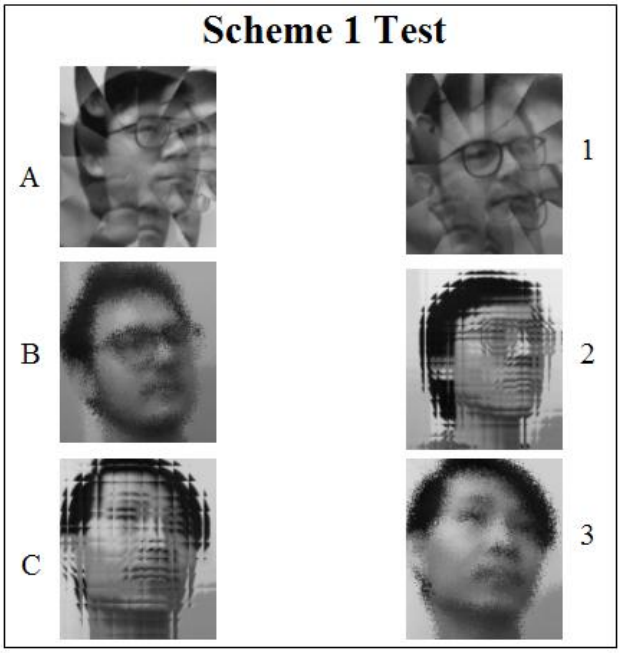


Figure 2.6: Example of the Misra and Gaj CAPTCHA [101].

ARTiFACIAL [21] asks users to identify the location of the eyes and mouth on a natural-looking face that is presented among a series of visually perturbed faces. While the CAPTCHA achieved a 78% human success rate, many users surveyed about ARTiFACIAL provided negative comments about its appearance [102]. Comments that the CAPTCHA was "eerie," "disturbing," and "creepy" helped halt its adoption [102, 18]. ARTiFACIAL is also vulnerable to attack, with Zhu et al. demonstrating an attack method with an 18% success rate [98].

Image orientation CAPTCHAs

Directcha [103] relies on user knowledge of how a pictured object should appear. Users are presented with an image of a rotated object. An entry pad is provided with several options

to indicate which direction the model is facing. In extremely limited testing with three participants attempting a set of eight images, Directcha achieved a 92% human success rate. The CAPTCHA's authors did not conduct testing with automated attackers, but a single instance could be expected to have a high 12.5% brute force attack success rate.

2.2.5 Video-based CAPTCHAs

Video-based CAPTCHAs ask users to watch short video clips rather than view images or text. Transmission of video imposes heavier bandwidth requirements on users compared to still images, which can make video-based CAPTCHAs ill-suited for use on mobile devices or through slow Internet connections.

Keyword-tagging CAPTCHAs

Keyword-tagging CAPTCHAs prompt users to provide descriptive keywords after watching a short video clip, similar to how ESP-PIX asks users to categorize still images [53]. Examples of these CAPTCHAs are depicted in Fig. 2.7. In *Kluever's Video CAPTCHA*[18, 104], users watch a video from YouTube and then type-in three descriptors. These descriptors are then compared against keywords specified by the video's original author using a keyword stemming and matching process. The large number of videos on YouTube provides a large source database, but the publicly accessible database also represents a potential attack vector if a would-be attacker is able to locate the matching video on YouTube. In testing, humans achieved a 90.2% success rate in solving Kluever's CAPTCHA [18] and automated attackers had a significant 13% success rate [104].

The *K L University CAPTCHA* [105] asks users to watch YouTube videos of commercial advertisements that have been modified to block out the name of the product being advertised. After viewing the CAPTCHA, the user is asked to select the type of product featured in the advertisement from a predefined list. This approach is highly attackable due to the small number of options (four) that users choose from for their answer.

(a) Kluever's Video CAPTCHA [105]

(b) KL University CAPTCHA [105]

Figure 2.7: Examples of keyword-tagging video CAPTCHAs.

Moving object recognition CAPTCHAs

An alternative video-based approach is to embed objects or text strings that move within a video as it plays. To solve the CAPTCHA, users identify these objects by clicking on them or typing in their text as appropriate. Cui et al. [106, 107] proposed a three-dimensional CAPTCHA based on moving text but never publicly disclosed a full implementation of their work. *NuCAPTCHA* [108] is a commercial implementation based on text that floats within a video image. While it has a 95% human success rate [109], the design choices enabling this high human success rate also make NuCAPTCHA easy to attack, thereby limiting its usefulness as a security tool. By exploiting the CAPTCHA's use of rigid character outlines, high contrast colors, non-overlapping movement trajectories, and reduced alphabet, Xu et al. achieved a 77% attack success rate [110].

2.2.6 Audio-based CAPTCHAs

Audio-based CAPTCHAs are commonly used as an accessible alternative to other forms of CAPTCHA for users with visual impairments. These CAPTCHAs impose additional system requirements compared to traditional visual CAPTCHAs since users are required to listen

to or record sound, which may not be possible on all types of computing devices or in all environments.

Listening CAPTCHAs

The most common form of audio CAPTCHA requires users to listen to a noisy or distorted sound clip and then type-in the words or numbers they hear. A number of implementations in this vein have been created by websites such as Authorize, Yahoo!, Google, eBay, Slashdot, and Digg [22], as audio-based alternatives for visual CAPTCHAs such as reCAPTCHA [111], and as standalone tests [112, 113]. While beneficial for accessibility purposes, this type of CAPTCHA is not secure. A number of successful attack strategies have been developed against this form of CAPTCHA involving filtering out added noise then analyzing the remaining sounds with voice recognition software [114, 115, 111]. Given the widespread use and advanced stage of development of voice recognition software, it is expected that this type of software would be able to accurately recognize the speech from these CAPTCHAs [116].

Meutzner et al. [117] and Yamaguchi and Kikuchi [118] have both proposed CAPTCHAs where users listen to computer-generated audio of words being read, but instead of adding noise or distorting the audio to deter attacks, these CAPTCHAs include nonsense word-like syllables intermixed with actual words. To solve these CAPTCHAs, users must correctly identify the real English words while ignoring non-word sounds. These solutions demonstrated poor human success rates in testing: Meutzner achieved only 29% accuracy when evaluating sentences and one tested configuration of Yamaguchi's scheme had a low 56% success rate.

Natural sound CAPTCHAs

The *Human-Interaction Proof, Universally Usable* CAPTCHA (HIPUU) [119] is designed to accommodate both sighted and non-sighted users. It displays an image while simultaneously

playing a sound effect appropriate for that image (e.g., the sound of thunder plays with a picture of lightning, or a bird sings while a bird is displayed on-screen). Users then choose a descriptor from a dropdown list corresponding with the image and sound. Despite the fixed number of options available in the dropdown list, HIPUU achieved only a 46% human success rate in testing. Completing the CAPTCHA using the audio-based method is a slow process, taking 65.6 seconds on average. Ryu et al. [120] propose a similar CAPTCHA that asks users to listen to a sound and then select a pictogram corresponding to the sound. No user testing was conducted, but its use of a limited number of fixed options suggests it is vulnerable to random guess-based attacks.

The *SoundsRight CAPTCHA* [121] asks users to identify when a certain type of sound (e.g., rooster crowing, crickets chirping) occurs in a played audio clip. Users are told the item to listen for before the clip plays and use the spacebar on their keyboard to denote when the sound starts and stops. The times they select for the beginning and end of the sound are compared to prerecorded values to determine if the user is human. This approach achieves a 92% success rate but its authors note that it is limited by the small number of sounds which humans are able to readily distinguish. Since this CAPTCHA's test only involves sound, it must be paired with a visual CAPTCHA to accommodate deaf users.

Audio analysis CAPTCHAs

Rather than asking users to identify sounds or recognize words and numbers, the *HuMan* CAPTCHA [122] requires participants to analyze audio recorded in public settings such as train stations and sporting games. Users are asked to type-in answers to context-sensitive questions based on the played audio, such as when a specific train is departing. Users can personalize the type of tests they are given through configurable options in the CAPTCHA's user interface or they can be selected at random. In testing with randomly selected questions, the average time to completion was over 35 seconds; personalized questions were answered somewhat faster, in 23-25 seconds. Study participants who were non-sighted were slightly

faster on average than sighted users in completing the tests. The average success rate was approximately 92%. HuMan's authors believe the CAPTCHA is resilient against attack due to the difficulty of recognizing the sounds in the audio and then answering context-sensitive questions, but the ability for users to manually customize the types of questions represents a significant security weakness.

Recording CAPTCHAs

Reading CAPTCHA [123] inverts the normal process used by audio CAPTCHAs; instead of having users play audio, they are required to record it. Users are provided with a passage of text to read. Their voice is recorded and uploaded to the CAPTCHA server, which analyzes it to determine if it was a natural human reading or a machine-generated voice. Reading CAPTCHA's authors claim a 97% success rate for humans with a 4% attack success rate, but this CAPTCHA has practical usability shortcomings since many desktop computers do not have microphones. It would also be difficult to obtain a clear recording in a public or noisy environment.

2.2.7 Interactive CAPTCHAs

Several CAPTCHA designs have been developed which require users to manipulate elements of the CAPTCHA. While these approaches can be more secure than other implementations, many have significant usability limitations.

Drag-and-drop CAPTCHAs

Several interactive CAPTCHAs employ tests requiring users to complete drag-and-drop actions to solve the CAPTCHA. *MosaHIP* [124], *Drag and Drop CAPTCHA* [125], *MOVTCHA* [126], and *Interactive CAPTCHA* [127] direct users to drag and drop items onto prescribed objects hidden within a larger composite image. *Zhang's CAPTCHA* [128] instructs users to drag the picture matching a specific object into a special drop area. Mobile devices, such

as smartphones and tablets, currently lack consistent support for drag-and-drop actions, rendering these CAPTCHAs unusable [129].

Visualization and orientation CAPTCHAs

Sketcha [130] incorporates line drawings based on three-dimensional models of real-world objects. Users are shown a series of 3-D model sketches, each presented from a random viewpoint and rotated to one of four increments. To solve the CAPTCHA, users must rotate the images to their original upright position. In testing, humans had a 78% success rate in solving an individual image but scale vector machine-based attackers had 61% accuracy. Sketchas normally contain eight images which must be rotated. This limits the automated attack success rate to 1.9% but also results in an extremely low 13.7% human success rate [130].

RotateCAPTCHA [131] presents users with a photograph divided into two concentric circles. Users solve the CAPTCHA by correctly rotating the contents of both circles to their upright position, aligning them to each other in the process. This CAPTCHA achieved a 71% human success rate in testing. Its authors do not believe an automated attack is likely, but it seems that an approach based on repeated rotations of each component and checking for the alignment of embedded lines could have reasonable success.

CAPTCHaSTAR [132] displays a dynamic image representing a star field. As users move their mouse cursor or finger on a touchscreen, the stars realign. Users are instructed to continue moving their pointer until the stars align to form the outline of an image. This process is somewhat slow; in testing, users took between 15 and 60 seconds, on average, depending on the difficulty of the CAPTCHA. Human success rates ranged from 46% to 91%. Attacks using support vector machine classifiers achieved a 78% success rate.

Dynamic cognitive game CAPTCHAs

Dynamic cognitive game (DCG) CAPTCHAs combine drag-and-drop or other dynamic movement with puzzle-like games [45]. By presenting the test as a sort of game, DCGs aim to reduce the resistance many human users have to using CAPTCHAs. In [45], Mohamed et al. propose games based on matching shapes, feeding animals, parking boats, and placing ships on a sea. Yang et al. [133] developed tests based on rolling balls and simulating the game Dance Dance Revolution. Several commercial implementations also exist, such as *PlayThru* [134], which asks users to complete tasks such as building a hamburger. While users can solve these tasks quickly and accurately in testing, they suffer from successful attack strategies approaching 100% accuracy [135]. DCGs also commonly have usability difficulties on mobile devices due to use of drag-and-drop input methods and Adobe Flash [136].

2.2.8 Problem-based CAPTCHAs

Due to the usability issues found in many traditional CAPTCHAs, some websites eschew them in favor of logic or language questions.

Mathematics CAPTCHAs

Several CAPTCHA approaches require users to complete mathematics problems in lieu of recognizing text or images. *QRBGS CAPTCHA* [46] is one example. It asks users to solve problems based on simple mathematical operations, such as multiplication and division, as well as more challenging tasks like solving for algebraic variables and calculating derivatives. The human success rate for this CAPTCHA are not available, but a note displayed with the CAPTCHA test suggesting users reload the page to obtain an easier test suggests it may not be high. Automated attackers achieved an overall success rate of 44.5%, which may represent an unusual case where computers are more successful at solving a CAPTCHA than humans. Other mathematics CAPTCHAs tend to use easier tests. For example, *Securimage*

[57], while primarily a text-based CAPTCHA, includes an option that generates tests based on simple addition, subtraction, multiplication, and division problems.

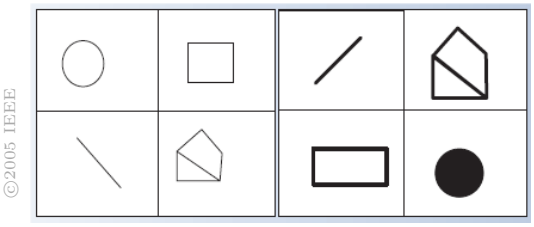
Language analysis CAPTCHAs

The *Strangeness in Sentences CAPTCHA* (SS-CAPTCHA) [137] presents users with a series of English sentences, some of which were written by humans and others which were created by machine translation and include non-standard language. Users are asked to identify which sentences seem natural and which appear "strange." In testing, humans achieved a 90% success rate in categorizing natural versus strange sentences. By the nature of its design, SS-CAPTCHA is language-dependent; non-native speakers of the language used in its tests are likely to encounter difficulty in correctly identifying if the sentence is strange or not. SS-CAPTCHA's authors note it is subject to attacks based on search engines and re-translation convergence but they do not quantify the exact likelihood of these attacks succeeding.

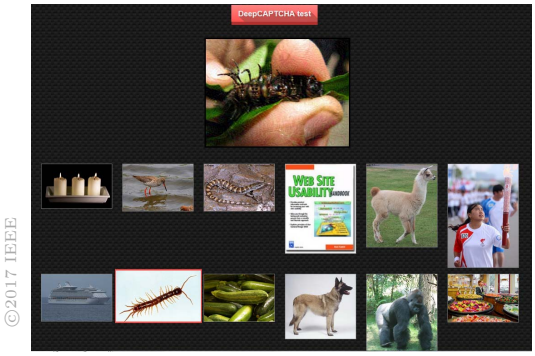
Semantic relationship CAPTCHAs

Several text- and image-based CAPTCHAs have been developed that require users to understand semantic relationships between items. Examples of this form of CAPTCHA are depicted in Fig. 2.8. *Bongo* [16] presents users with two sets of images, each set containing images with some common visual characteristic such as having bold lines versus thin lines or solid shapes versus hollow shapes. The user is provided with a single image and asked to identify the set to which it belongs. As there are two possible choices, a single Bongo instance has an inherent 50% random guess success rate. To reduce its vulnerability to random guess-based attacks, multiple Bongo instances can be used together [138]. This, however, requires a substantial amount of screen space or that the user complete multiple rounds of tests.

SEMAGE [141] presents users with a set of images and asks them to recognize relationships between the images by choosing those which are of the same type of object. Users



(a) Bongo [138]



(b) DeepCAPTCHA (2017) [139]

(c) SemCAPTCHA [140]

Figure 2.8: Examples of semantic relationship CAPTCHAs.

achieved a 94.8% success rate in testing. No attack analysis has been completed on this CAPTCHA, but it appears to be vulnerable to simple image classifier-based attacks due to its use of discrete images. *DeepCAPTCHA* (2017) [139] functions similarly, asking users to select objects matching a presented example. To deter image classifier-based attacks, the presented example image is visually distorted with immutable adversarial noise computed by a deep learning algorithm. While the CAPTCHA achieved an 82.6% human success rate, it remains vulnerable to brute force attacks. DeepCAPTCHA's authors recommend requiring users complete at least two CAPTCHAs as part of the verification process.

SemCAPTCHA [140] takes the opposite approach to SEMAGE and DeepCAPTCHA (2017). It presents users with rendered text of various object names and asks users to identify which one does not belong. For example, users might be asked to choose which item does not fit from ducks, cuckoo birds, and cows. Again, humans performed well, solving this

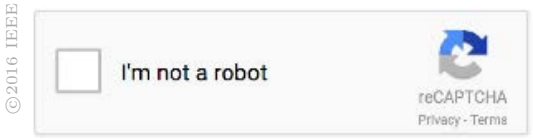
CAPTCHA in under 5.5 seconds on average. No automated attack rates are available, but an attack seems trivial to complete given the availability of highly accurate optical character recognition and the small number of words embedded in each SemCAPTCHA image.

Another CAPTCHA named *DeepCAPTCHA* [47], released in 2014 by Nejati et al. and unrelated to the 2017 CAPTCHA of the same name, presents users with a set of similarly sized images representing various objects. Using their knowledge of depth perception and relative sizes, users are asked to order the images by the size of the real world objects they depict. It achieved an 87.7% human success rate in testing. While its authors claim a low automated attack success rate, its use of discrete images makes its vulnerable to highly accurate image classifiers.

2.2.9 Behavior analysis-based CAPTCHAs

Another approach to avoiding the challenges posed by increasingly sophisticated OCR and image recognition technology is to make decisions on whether or not a user is human based on their recorded behavior. Google has been a leader in this area. Their *reCAPTCHA v2* (also known as *No CAPTCHA reCAPTCHA*) [48] system presents users with a checkbox labeled "I'm not a robot" as shown in Fig. 2.9. Users click or tap this checkbox to certify they are human. If reCAPTCHA's risk analysis system believes the user is human, they are granted access to the protected resource; if it does not, the user is required to complete a secondary image classification or text-based reCAPTCHA test [142]. *Invisible reCAPTCHA* [48], an alternative implementation, foregoes the initial "I'm not a robot" checkbox; when the risk analysis system is unsure if the user is human, the CAPTCHA goes straight to an image classification task.

Much of how these approaches determine if the user is actually human is not publicly known. Google indicates the CAPTCHAs make use of an "advanced risk analysis backend" to make this determination [144]. Researchers have identified that the risk analysis process includes evaluation of factors such as the user agent string, browser environment, mouse



(a) "I'm not a robot" checkbox

(b) Secondary image classification test

Figure 2.9: Samples of reCAPTCHA v2 (No CAPTCHA reCAPTCHA) [143].

movements when solving the CAPTCHA, and the existence of a sufficiently old Google tracking cookie [142, 145]. While these CAPTCHAs enable a smooth user experience, since most users complete either a simple CAPTCHA or no CAPTCHA at all [146], they have significant security vulnerabilities. Sivakorn et al. [143] have developed strategies to increase the likelihood that a user will be shown the "I'm not a robot" checkbox test, rather than a full CAPTCHA, and then to solve the checkbox test. In the event a secondary CAPTCHA is displayed, attacks can defeat the image classification task with 71% accuracy [143] and text-based tests with a 40% success rate [23]. Even robots have been shown to successfully complete reCAPTCHA v2's "I'm not a robot" test [147].

2.3 Summary

Since the advent of the Internet, significant effort has been dedicated to creating strategies to protect online systems from automated attack. CAPTCHAs are one of the most visible and heavily used security tools. Despite the creation of a multitude of approaches, few implementations successfully balance the need for ease of use by humans across a wide variety of platforms with resistance against automated attackers. In this research, we seek to ex-

plore new approaches that achieve these important goals by providing good user experiences without sacrificing security.

Chapter 3

Face Detection CAPTCHA

3.1 Proposed approach

3.1.1 CAPTCHA design

Face Detection CAPTCHA [148] is an initial approach designed to avoid the security and usability shortcomings of existing CAPTCHAs by using tests based on complex composite images. It presents users with an image that includes several embedded human and non-human faces. The faces are visually distorted and randomly placed on a noisy background. To successfully solve the CAPTCHA, users must correctly click on all genuine targets, the human faces, without any erroneous clicks. An example of Face Detection CAPTCHA marked to indicate the correct solution is shown in Fig. 3.1.

Face Detection CAPTCHA provides several benefits over existing approaches. Compared to text-based CAPTCHAs, this methodology avoids the escalating difficulty caused by improving OCR technologies. It also avoids potential language barriers since no text is used, making the design language-independent and thus deployable to a global audience. Since Face Detection CAPTCHA involves no text entry, it can easily be used on mobile devices that lack a convenient keyboard. Compared to existing image-based CAPTCHAs, this approach does not rely on small classification sets, like ESP-PIX, or display discrete images,

Figure 3.1: Sample Face Detection CAPTCHA image with correct answers circled in red.

which can allow a high likelihood of success for brute force attacks by automated algorithms. Human face detection is a complicated task for computers to successfully perform, especially when the faces are visually distorted and presented in complex composite images. Unlike previous face-based CAPTCHAs, Face Detection CAPTCHA uses natural-looking faces to avoid user concerns about eerie and disturbing images [102], and it presents the faces as part of a composite image to minimize the likelihood of a successful brute force attack.

3.1.2 Generation process

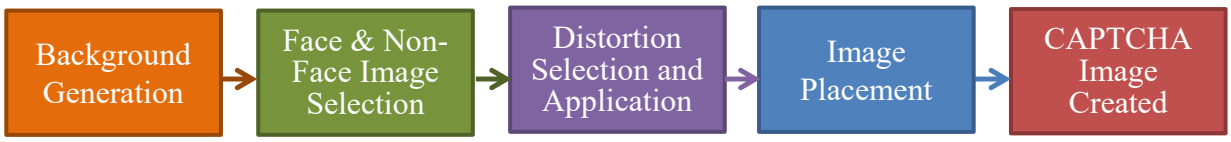


Figure 3.2: Steps involved in generating Face Detection CAPTCHAs. First, a complex background image containing many shaded rectangles is generated. Next, face and non-face images are selected for embedding, then distortions are chosen and applied to the face and non-face images. These distorted images are then placed on the background, yielding a completed Face Detection CAPTCHA image.

As shown in Fig. 3.2, creating Face Detection CAPTCHA images is a multi-step process. It can be represented as:

$$C = F(\phi, I_{genuine}, I_{fake}) \tag{3.1}$$

where function F creates a new CAPTCHA image containing embedded images taken from sets $I_{genuine}$ and I_{fake}. Distortion settings (distortion types and distortion intensities) selected from ϕ are applied to the rendered composite, yielding CAPTCHA C.

Background generation

Generation of a new Face Detection CAPTCHA image begins with the creation of a new 500×300 pixel background composed of overlapping grayscale rectangles of various sizes as shown in Fig. 3.3. The noisy background's pattern of overlapping shapes, colors, and lines is designed to thwart the effectiveness of using edge detection to identify the outline of embedded face images.

Figure 3.3: Sample of the rendered background [148].

Image selection

After the background is created, four to five genuine human faces and fake non-human faces are chosen to be embedded so that:

$$n_{total} = \left\{ n_{genuine} + n_{fake} \;\middle|\; n_{genuine} \geq 1, n_{fake} \geq 1, n_{total} = \{4, 5\} \right\} \tag{3.2}$$

where $n_{genuine}$, n_{fake}, and n_{total} represent the number of embedded genuine, fake, and total images, respectively. At least one embedded image is genuine and another is fake. Each

embedded image is converted to grayscale and scaled to a randomly selected size between 50×50 and 100×100 pixels.

Distortion selection and application

To increase the difficulty for a would-be automated attacker to solve Face Detection CAPTCHA, each generated CAPTCHA is visually distorted. For the CAPTCHA, one distortion type is selected at random from Table 3.1 to be applied to each embedded image. Each distortion type has one to three associated fixed intensity levels, and one intensity level is selected at random for use with the CAPTCHA image.

Table 3.1: Face Detection CAPTCHA distortion types and intensities [148]

Distortion Used	**Parameters Adjusted**	**Intensity Level**		
		Low	**Medium**	**High**
Blurring	Standard deviation	4	8	20
Closing	Radius	3	5	-
Erosion	Radius	3	-	-
Laplacian filtering	a	-	10	5
Lighten	Histogram max-range	-	-	0.3
Periodic noise	% of image removed	67%	-	-
Piecewise scaling	Scale factor	2:1	3:1	-
Resolution modification	Scale factor	1:4	1:8	1:10
Rotation	Degrees rotated	-	90	180
Width scaling	Scale factor	4	5	-
Height scaling	Scale factor	2.5	3	4
Speckle noise	Variance	0.2	1	-
No distortion	-	-	-	-

As an example, the lighten distortion can be applied to an image using the equation:

$$v'(x,y) = \left\{ \min\left(v(x,y) + 255\varphi, 255\right) \;\middle|\; 0 < \varphi < 1 \right\} \tag{3.3}$$

where $v(x, y)$ is the original value of the 8-bit grayscale pixel at coordinates (x, y). φ is the intensity level for the lighten distortion. $v'(x, y)$ is the resulting value for pixel (x, y) after the distortion has been applied. The images shown in Fig. 3.4 model the lighten distortion when applied to the Lena image [149].

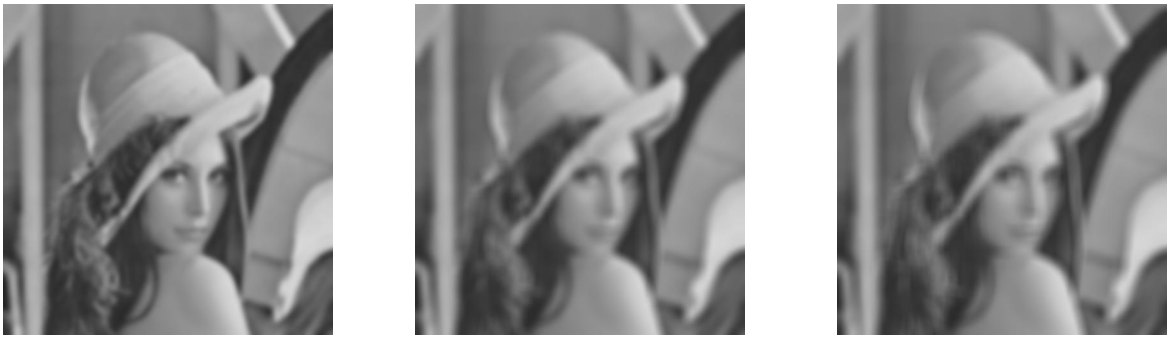
(a) Undistorted image (b) $\varphi = 0.6$ (c) $\varphi = 0.3$

Figure 3.4: Lena image with applied lighten distortions.

Another example of distortion is blurring using the 2D Gaussian equation [150]:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{3.4}$$

where x represents the horizontal distance from the origin, y represents the vertical distance from the origin, and σ is the standard deviation. When the Gaussian blur distortion is applied to the Lena image, the resulting distorted images are displayed in Fig. 3.5.



(a) $\sigma = 4$ (b) $\sigma = 8$ (c) $\sigma = 20$

Figure 3.5: Lena image with applied Gaussian blur [148].

Image placement

Once each embedded image has been distorted, locations are randomly chosen to place each image on the background. The distorted face images are randomly placed so that no two images overlap with the other. Fig. 3.6 shows examples of final CAPTCHAs generated with the process described.

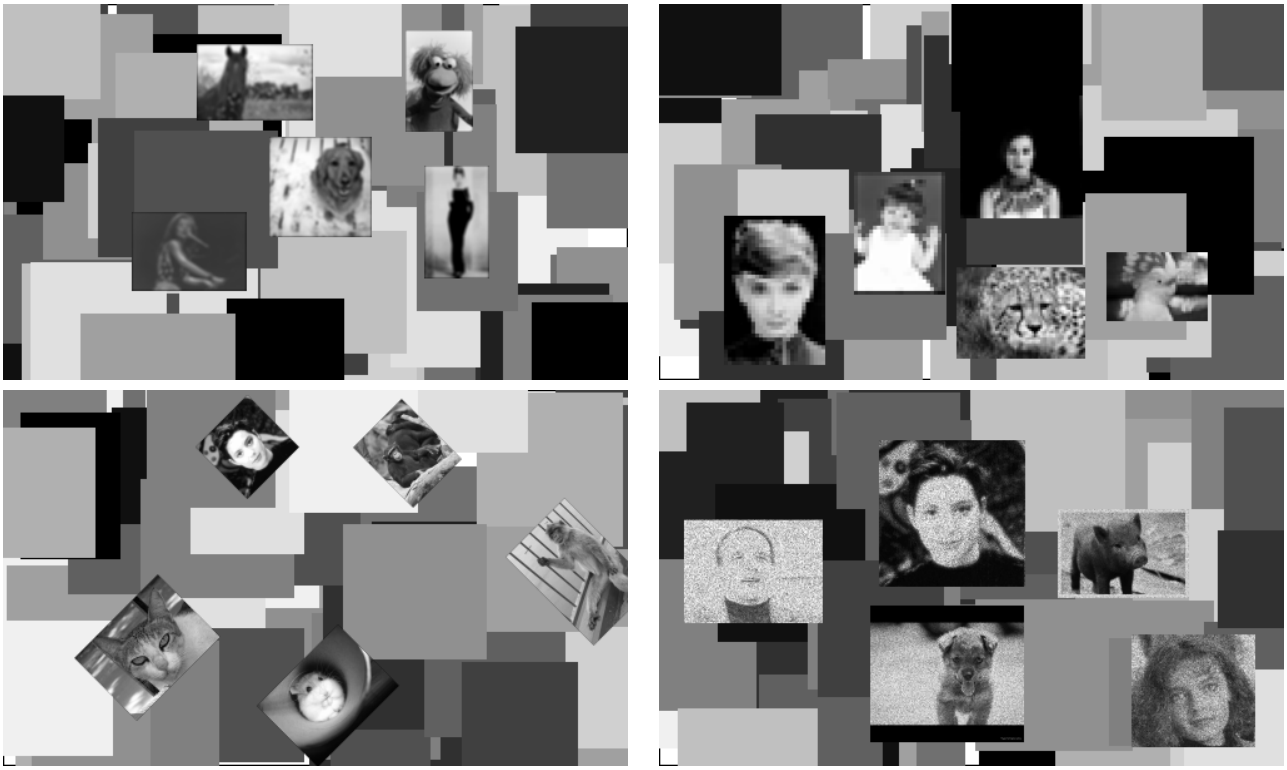


Figure 3.6: Samples of generated Face Detection CAPTCHA images.

3.2 Experimental results and analysis

Face Detection CAPTCHA images underwent testing by 1,100 participants. This section provides details on the source images, research participants, and protocols used in evaluating the CAPTCHA along with results and analysis.

3.2.1 Image databases

The embedded images used in Face Detection CAPTCHA were sourced from publicly available online images. Genuine human face images were chosen from a subset of Carnegie Mellon University's Front Face Images database [151]. Fake non-human face images were taken from photographs of animals and puppets posted on Flickr [152].

3.2.2 Participants and testing protocol

To evaluate human accuracy in solving Face Detection CAPTCHAs, a group of 1,100 individuals was asked to access a website protected by the CAPTCHA. Users were unsupervised and allowed to access the website using their choice of browser and computing device. For each attempt, the CAPTCHA Evaluation Platform (see Chapter 9) was used to present a CAPTCHA newly selected from a set of 1,156 images. Users were asked to continue attempting to solve CAPTCHAs until they were successful.

Automated attacks against the CAPTCHA were simulated using the Viola-Jones face detection algorithm [153]. This algorithm works by calculating the integral image, the sum of all pixel values to the left and above a given point, as shown by:

$$ii(a,b) = \sum_{a' \leq a, b' \leq b} i(a', b') \tag{3.5}$$

Here, a, b are points, $i(a,b)$ is the original image, and $ii(a,b)$ is the corresponding integral image. Using the integral image, a series of Haar-like rectangular features are computed across the image. The rectangular features are run through a cascade of classifiers to determine the probable locations of embedded faces [153, 154].

3.2.3 Analysis

Human performance evaluation

In total, 8,995 attempts at solving Face Detection CAPTCHA were recorded with an overall average success rate of 71.8%. As shown in Table 3.2, human success rates varied significantly depending upon the type of distortion applied. When no distortion was applied, the human success rate was 82.48%. When distortions were applied, rates varied from 81.4% to 64.1% depending upon the distortion type used. Distortions which introduced noise or applied blurring-type effects on the image had the best human success rates. Those which adjusted contrast or scaled the image, particularly image widths, had the lowest human success rates.

Automated attack evaluation

Just as with human success rates, automated attack success rates varied greatly depending upon the distortion type used. Viola-Jones successfully solved 16.7% of undistorted CAPTCHAs. Automated attack success rates varied from 0% to 16.7% on distorted CAPTCHAs, with an overall average success rate of 9.3%. In general, distortions that were challenging for human users were challenging for automated attackers and vice versa. Results for specific distortion types are shown in Table 3.2.

While there were cases where automated attacks were unable to correctly detect the human faces in a CAPTCHA image, the basic design of a Face Detection CAPTCHA requires that at least one genuine human face must be present. If an automated attacker is unable to detect any face, it may attempt to solve the CAPTCHA by guessing the location of the genuine faces. If the guess is done completely at random, the likelihood of defeating the CAPTCHA is remote. Assuming a CAPTCHA with five embedded images (one to four of

Table 3.2: Human and automated attack success rates by distortion type for all intensity levels [148]

Distortion Used	**Human Success**	**Attack Success**
No distortion	82.5%	16.7%
Periodic noise	81.4%	14.3%
Resolution modification	80.3%	15.1%
Erosion	79.5%	16.7%
Rotation	79.4%	1.6%
Blurring	79.0%	16.7%
Speckle noise	79.0%	13.1%
Piecewise scaling	77.4%	7.1%
Closing	75.9%	11.9%
Height scaling	75.4%	0.0%
Lighten	68.7%	9.5%
Laplacian filtering	66.3%	0.0%
Width scaling	64.1%	0.0%
Overall	**71.8%**	**9.3%**

which are genuine faces) and an average face bounding box size of 48×51 pixels as in the test CAPTCHAs, the average chance of a correct guess is 4-in-1000:

$$\left(\frac{1}{4}\right)\sum_{i=1}^{4}\left(\prod_{j=1}^{i}\frac{(48\times 51)j}{(500\times 300)-(i-j)}\right)=0.422\% \tag{3.6}$$

Comparing success rates to image quality metrics

To better understand the relationship between distortions, human success rates, and automated attack success rates, the structural similarity (SSIM) [155] and visual information fidelity (VIF) [156] image quality metrics were used to compare distorted and undistorted versions of each CAPTCHA. These metrics are patterned after the human visual system and objectively measure differences between images. Both metrics yield values in the range $[0, 1]$, where values closer to 1 represent images that are more visually similar. As shown

in Table 3.3, there is a correlation ($r = 0.48$) between average human success rates and average SSIM values for each distortion type. There is also a correlation ($r = 0.88$) between average automated attack success rates and average VIF values as shown in Table 3.4. These relationships suggest the SSIM and VIF metrics are useful in predicting human and automated attacker performance in solving CAPTCHAs. This information can be leveraged to adjust CAPTCHA generation settings in future implementations to create better-performing CAPTCHAs.

Table 3.3: Comparison of human success rates and SSIM values [148]

Distortion Used	**Human Rank**	**SSIM Rank**	**Human Success**	**SSIM Value**
Periodic noise	1	8	81.4%	0.8161
Resolution modification	2	2	80.3%	0.9608
Erosion	3	4	79.5%	0.9303
Rotation	4	11	79.4%	0.7468
Blurring	5	3	79.0%	0.9403
Speckle noise	6	5	79.0%	0.9146
Piecewise scaling	7	8	77.4%	0.7609
Closing	8	1	75.9%	0.9677
Height scaling	9	9	75.4%	0.7540
Lighten	10	6	68.7%	0.8556
Laplacian filtering	11	12	66.3%	0.7293
Width scaling	12	10	64.0%	0.7526

Considerations for future implementations

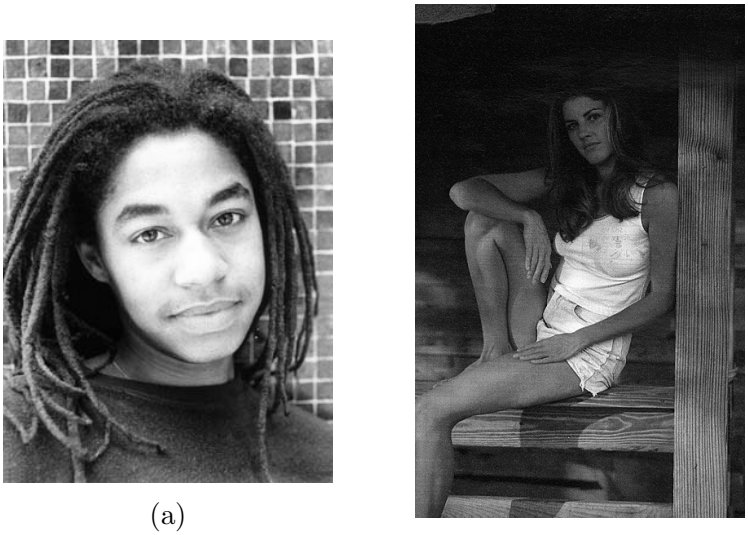
Experimental results show that several factors must be taken into consideration when implementing CAPTCHAs to ensure an optimal balance between human and automated attack success rates. One of the most critical choices is selecting the embedded genuine face images. Attention should be given to the pose, size, illumination, and facial expression of the

Table 3.4: Comparison of automated attack success rates and VIF values [148]

Distortion Used	**Attack Rank**	**VIF Rank**	**Attack Success**	**VIF Value**
Erosion	1	5	16.7%	0.5799
Blurring	2	4	16.7%	0.6239
Resolution modification	3	1	15.1%	0.7276
Periodic noise	4	6	14.3%	0.5574
Speckle noise	5	3	13.1%	0.6342
Closing	6	2	11.9%	0.7071
Lighten	7	7	9.5%	0.4983
Laplacian filtering	8	8	0.0%	0.3705
Piecewise scaling	9	9	7.1%	0.3579
Rotation	10	10	1.6%	0.3433
Height scaling	11	11	0.0%	0.3376
Width scaling	12	12	0.0%	0.3327

subject as these factors significantly impact a human's ability to recognize the face once it is distorted and embedded in the final composite CAPTCHA. As an example, consider the two images in Fig. 3.7 that were paired together in 49 different rendered CAPTCHAs.

Fig. 3.7a is a bust shot displaying good contrast. Humans correctly identified it as a human face 99.0% of the time. By comparison, Fig. 3.7b is dark, has poor contrast, and the face comprises a relatively small portion of the full image. People only identified it in 61.8% of attempts. Effort should be given to removing items similar to Fig. 3.7b that are unlikely to perform well from the source image set, but there may be further value in using a standardized image set such as a driver's license photograph database. Standardized images with a neutral pose, expression, and illumination will help to ensure the effect of applying distortions is predictable while also providing a consistent appearance for the human users attempting to recognize the human faces. The large variation in the CMU face database images used is reflected in lowered human detection rates.



(a) (b)

Figure 3.7: The original face image can have a significant impact on how well humans are able to identify it once embedded [148].

The two other key parameters to ensure optimal CAPTCHA effectiveness are the distortion type and distortion intensity applied to embedded images. During initial development of Face Detection CAPTCHA, 24 different distortion types were considered. After reviewing their effects when applied to images and their differences from other distortions, only 12 were selected for use in the final CAPTCHA implementation. Based on the experimental results, several observations can be made regarding the performance of different distortions types and their value. Geometric distortions such as scaling and rotating images functioned best, with noise-based distortions also of significant benefit. Mathematical morphologies such as closing and erosion have a mixed value; they perform well for human detection but also have above-average automated attack detection rates. Distortions impacting the contrast ratio are of limited use due to high automated attack success rates relative to human performance.

One or more intensities must be selected for each distortion type. It is beneficial to test several different intensities to identify the intensities that work best with a given image set. Depending on the underlying characteristics of the images, certain values may not be appropriate. For instance, brightly illuminated source images that then have the lighten distortion applied may appear completely washed out. After testing several different pa-

rameter settings, it is frequently possible to divide the intensities into a few similar groups by their relative level of effect. In the testing, the selected distortion intensities coalesced towards high, medium, and low settings; with further evaluation, some specific intensities were removed because they provided little difference or resulted in unusable images.

While selecting standard distortion types and intensities to apply to all images is a viable means of generating CAPTCHAs, as demonstrated by the results here, other options may be better. To achieve the best possible performance, the distortion type and intensity can be customized for each individual embedded image. The selection process may be automated by analyzing image quality metrics. In the experiments, SSIM and VIF were used to model how well humans and computers, respectively, could identify each image. By creating a composite of these or other image quality metrics, one can easily compare a large number of distortion parameters to find those with an optimal balance between human and automated attack detection accuracies. Similarly, embedded images can be compared to remove images likely to perform poorly. In this way, when the CAPTCHA image is generated, we can know it has the best possible likelihood of preventing automated attacks with minimal inconvenience to people.

3.3 Summary

Face Detection CAPTCHA represents an initial attempt at creating an image-based CAPTCHA using object detection as its test. While testing shows the CAPTCHA's human and automated attack success rates to be less than desirable for a publicly used CAPTCHA, the evaluation process demonstrates the viability of the basic concept and identified factors to improve the performance of future CAPTCHA designs.

Chapter 4

FaceDCAPTCHA

4.1 Proposed approach

4.1.1 CAPTCHA design

Even after decades of research in face detection, challenges remain for automated algorithms in accurately recognizing distorted or partially occluded faces in complex images. Humans, conversely, are skilled at recognizing faces in these situations. *FaceDCAPTCHA* [157] leverages these relative differences as it builds upon Face Detection CAPTCHA's image-based CAPTCHA framework.

FaceDCAPTCHA presents users with a composite color image containing four to six embedded images of human faces, cartoon faces, or sketches as shown in Fig. 4.1. As with Face Detection CAPTCHA, users solve FaceDCAPTCHA by selecting all of the embedded human face images without any mis-selections. Compared to its predecessor, FaceDCAPTCHA offers significant improvements that are designed to improve human success rates while reducing the likelihood of successful automated attacks. These improvements include FaceDCAPTCHA's use of facial feature occlusions, which hide elements such as the eyes, nose, and mouth that automated algorithms use to detect faces. FaceDCAPTCHA also employs a gradient descent optimization process to select distortion types and intensities enabling the

highest success rates for human users. These improvements result in a 15-point increase in human success rates while reducing automated attack success rates to nearly zero.

Figure 4.1: Sample FaceDCAPTCHA image with correct answers circled in red.

4.1.2 Generation process

The FaceDCAPTCHA generation process can be written as:

$$C = F(\phi, I_{genuine}, I_{fake}) \tag{4.1}$$

where function F represents a series of image processing operations applied using a set of distortion parameters, ϕ, and source images, $I_{genuine}$ and I_{fake}, to generate FaceDCAPTCHA image C. The set of parameters is used to control the difficulty of the generated CAPTCHA. FaceDCAPTCHA uses an adversarial learning approach and gradient descent optimization learning to learn the optimal set of parameters that maximize the human success rate at solving the CAPTCHA while minimizing the automated attack success rate. The FaceDCAPTCHA generation process can be divided into two phases: (1) creation of and performance collection using training CAPTCHAs, and (2) generation of the final FaceDCAPTCHA images using optimized parameters.

4.1.3 CAPTCHA generation parameters

Several CAPTCHA generation parameters in ϕ are chosen using results from an adversarial learning process:

- Variable n_{total} determines the total number of embedded images, including both genuine and fake faces. Genuine faces are images of real humans collected from different publicly available face databases. Fake faces are images of cartoons and other objects known to generate false positives by automatic face detection algorithms.

- The number of genuine face images in a CAPTCHA, $n_{genuine}$, is the second parameter. In a given CAPTCHA,

$$n_{total} = \left\{ n_{genuine} + n_{fake} \;\middle|\; n_{genuine} \geq 2, n_{fake} \geq 1, n_{total} = \{5, 6\} \right\} \tag{4.2}$$

  where n_{fake} is the number of fake images. For a given CAPTCHA, only n_{total} and $n_{genuine}$ must be defined.

- The CAPTCHA background, controlled by parameter set $\mathbf{B}$, is important in providing randomness to confuse automatic face detection algorithms. This parameter set specifies settings such as the number of background shapes to be generated (n_{shapes}), the number of dilation operations to be performed ($n_{dilations}$), and the number of random portions to be placed ($n_{portions}$).

- The (x, y) location of each embedded image is an important factor. With random locations, segmentation is more difficult than if a fixed location scheme is used.

- Parameter set $\mathbf{D}$ specifies settings for the five types of distortion operations that can be applied to the CAPTCHA. The distortion types available include stripes, strikeouts, rotation, background blending, and noise addition.

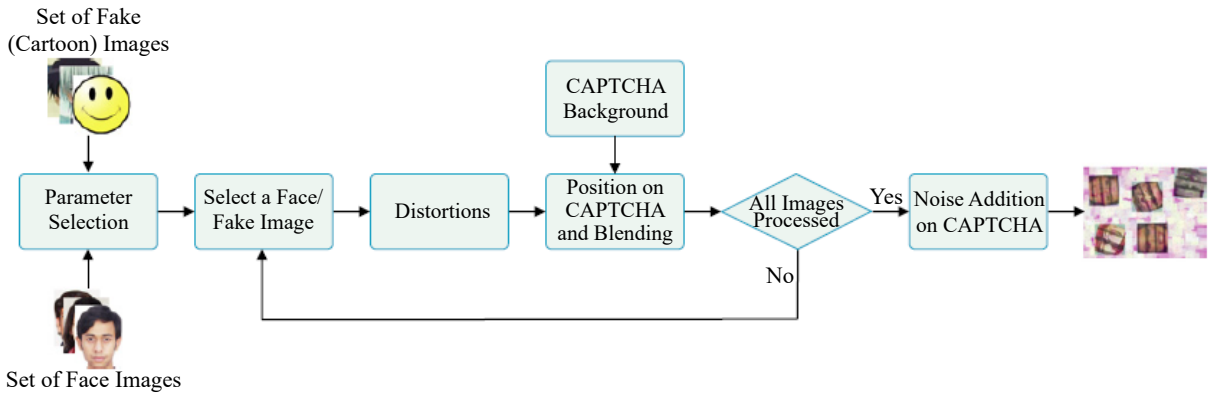


Figure 4.2: Steps involved in generating FaceDCAPTCHA images [157]. Human face and fake distractor images are selected and then visually distorted. They are placed on a complex multicolored background. Once all images have been placed, additional noise is added to the image, yielding the final FaceDCAPTCHA image.

FaceDCAPTCHA's gradient descent optimization process uses results from human users and automated attack adversaries attempting to solve training CAPTCHAs to find optimal values for the ϕ generation parameters. Training images were generated using values from 17 predetermined sets covering a wide range of possible values.

Both training and final FaceDCAPTCHA images are generated using the process shown in Fig. 4.2. It is described below.

Image selection

Once the generation parameters are determined, creation of the CAPTCHA image begins by selecting $n_{genuine}$ and n_{fake} images from sets $I_{genuine}$ and I_{fake}, respectively.

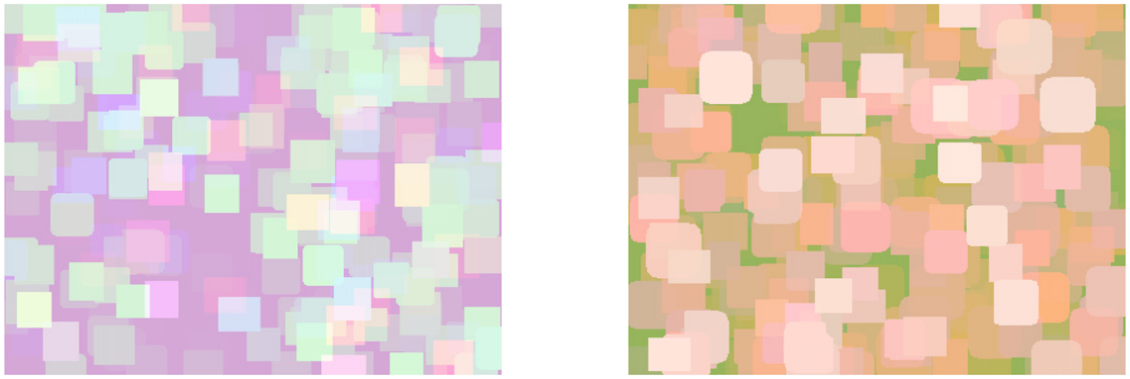
Background generation

Once the embedded images are selected, the generation of the 400 × 300-pixel background image begins using the following two approaches depending upon the specified values of n_{shapes} and $n_{portions}$:

- Random Colors: When $n_{portions} = 0$, the background image is created using shapes such as circles, squares, and crosses with randomly chosen sizes and colors. These shapes are

pasted on the canvas at random coordinates to generate the final background image. This background image is then dilated $n_{dilations}$ times before genuine and fake images are embedded to blur the edges of the shapes and confuse edge detection-based attack algorithms. Fig. 4.3a shows a sample background generated using this approach.

- Random Portions: When $n_{portions} > 0$, portions of the selected genuine embedded images are incorporated into the background. After generating an initial background image using the Random Colors algorithm, skin tone-colored sections of the genuine images to be embedded are randomly selected. These skin tone portions are placed on the background at random coordinates to help fool skin color-based face detection algorithms. Fig. 4.3b shows a sample background generated using this approach.



(a) Random Colors approach (b) Random Portions approach

Figure 4.3: Backgrounds generated using the Random Colors and Random Portions approaches [157].

Image-level distortion application

Each genuine and fake image to be embedded is processed using the distortion parameters specified by $\mathbf{D}$:

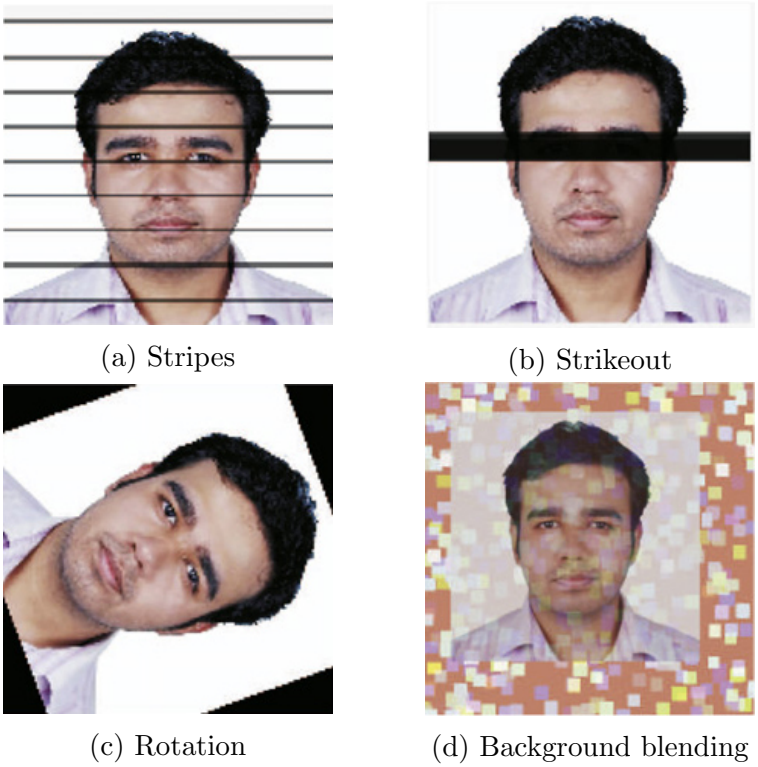
- Stripes are applied on some genuine and fake embedded images. This operation is not necessarily applied uniformly to all embedded images. A total of $\boldsymbol{D_{1N}}$ stripes are

applied, with the stripes being $\boldsymbol{D_{1W}}$ pixels in width, where $3 \leq \boldsymbol{D_{1W}} \leq 6$. An example of this operation is shown in Fig. 4.4a.

- The strikeout operation, shown in Fig. 4.4b, is used to occlude key facial features such as the eyes, mouth, and/or nose. The transparency parameter, $\boldsymbol{D_2}$, is set so that human users can visualize the facial features but it is challenging for adversaries.

- The rotation operation [158] is used to rotate genuine and fake images with a θ° angle as shown in Fig. 4.4c. $\boldsymbol{D_3}$ contains the θ angles for each embedded image.

- The blending operation [159] is used to smoothly blend the embedded genuine and fake images into the background as shown in Fig. 4.4d. In this operation, parameter D_4 is to determine the degree of blending.



(a) Stripes (b) Strikeout

(c) Rotation (d) Background blending

Figure 4.4: Effects of distortion operations on embedded images [157].

Image placement

Once each embedded image has been distorted, they are placed at randomly selected locations (x, y) on the CAPTCHA background $\mathbf{B}$. The embedded images are positioned to avoid overlaps with the edge of the CAPTCHA and with each other.

Global noise distortion application

Once the embedded images have been placed on background $\mathbf{B}$, noise-based distortions are applied to the entire CAPTCHA. The parameter n_{type} determines if additive, multiplicative, or salt-and-pepper noise is to be added. n_{min} and n_{max} determine the minimum and maximum percentage of pixels to be affected by the noise operation.

CAPTCHA parameter training

Training CAPTCHAs generated using predetermined parameter value sets are used to learn which parameter sets result in CAPTCHAs that are readily solvable by humans but not by automated attack adversaries. These useful parameter sets, ϕ_u, are learned using:

$$\phi_u = Train_{(C=F(\phi_i, I_{genuine}, I_{fake}))}(H = 1, Ad = 0) \tag{4.3}$$

where H is the human response and Ad is the adversary response to solving the CAPTCHA. Here, $H = 1$ represents the correct human response and $Ad = 0$ represents the incorrect response of the adversary. There can be several available parameters, ϕ_i; however, only those parameters (ϕ_u) satisfying the condition given in Eq. 4.3 are selected. Parameters associated with other conditions, i.e., $(H = 1, Ad = 1), (H = 0, Ad = 1)$, and $(H = 0, Ad = 0)$, are not useful. $Train$ represents parameter learning in a gradient descent manner. Let $E_{H,Ad}(\phi_t)$ be

the objective or error function that minimizes the error caused by four conditions associated with H and Ad:

$$E_{H,Ad}(\phi_t) = \begin{cases} 0 & \text{if } H = 1 \text{ and } Ad = 0 \\ 1 & \text{otherwise.} \end{cases} \tag{4.4}$$

In gradient descent optimization, optimal parameters are obtained using Eqs. 4.5 and 4.6:

$$\bigtriangledown E_{H,Ad}(\phi_t) = \frac{\partial E_{H,Ad}(\phi_t)}{\partial \phi} \tag{4.5}$$

$$\phi_{t+1} = \phi_t - \eta \bigtriangledown E_{H,Ad}(\phi_t) \tag{4.6}$$

Here, $\bigtriangledown E_{H,Ad}(\phi_t)$ represents the gradient of the error function at the tth learning iteration and η is the learning rate that controls the convergence of parameter learning. This learning procedure involves determining the human response (H) as well as the adversarial response for the automatic face detection algorithm (Ad); it converges to the case where humans can solve the CAPTCHA but the adversarial algorithm cannot.

Human responses were gathered from users attempting to complete the generated training CAPTCHAs. The Viola-Jones face detector [153], Picasa [160], and a commercial off-the-shelf face detection system (COTS) provided adversarial responses using the following methodology:

1. For each CAPTCHA image C, the Viola-Jones face detector [153] is first used to perform face detection.

2. The detected face coordinates are matched with the actual embedded genuine face photograph locations.

3. If all embedded faces in C were found, the face detector's adversarial response is considered correct. Otherwise, it is considered incorrect.

4. Steps 2 through 4 are repeated using Picasa in place of the Viola-Jones face detector, and then again with a COTS face detection solution to locate embedded genuine face images.

Final CAPTCHA generation using optimized parameters

Once the gradient descent optimization process is completed, the final FaceDCAPTCHA images are generated using the 10 best-performing optimized parameter sets as determined by Eq. 4.6. These CAPTCHAs are created using the same generation process described earlier in this section as is used for the training CAPTCHAs. Fig. 4.5 shows several examples of the final optimized FaceDCAPTCHA images.

Figure 4.5: Samples of optimized FaceDCAPTCHA images [157].

4.2 Experimental results and analysis

FaceDCAPTCHA was tested by over 1,300 participants. This section describes the source images used to generate the CAPTCHA, experimental protocol, and key results.

4.2.1 Image databases

The embedded images used in FaceDCAPTCHA were sourced from publicly available online images. Approximately 1,800 genuine human face images were chosen from the LFW face database [161] and about 300 fake non-face images were taken from cartoons posted on Photobucket [162]. The fake images were selected because they resulted in false positives when analyzed by the Viola-Jones face detection algorithm [153]. Sixty percent of the source images were used in generating training images; the remaining source images were used in generating the final optimized test CAPTCHAs.

4.2.2 Participants and testing protocol

To evaluate human accuracy in solving FaceDCAPTCHA images, over 1,300 participants attempted to access a website protected by a CAPTCHA. Users were unsupervised and allowed to access the website using a browser and computing device of their choice. They were asked to continue attempting to solve CAPTCHAs until they succeeded. Results were initially gathered on 5,300 training CAPTCHAs, and once the gradient descent optimization process was complete, the 500 final optimized CAPTCHAs were also tested.

Automated attacks against FaceDCAPTCHA final optimized images were simulated using the Viola-Jones algorithm [153], Picasa [160], and a commercial off-the-shelf face detection solution. A superset of 17,000 CAPTCHAs were evaluated by automated attackers during the training phase. The same 500 optimized CAPTCHAs as with human users were evaluated for final results.

4.2.3 Analysis

Training results

Of the 11,000 attempts at solving training FaceDCAPTCHA images, the overall human success rate was 66.8%. Human success rates for the various training sets varied from 64%

to 86%, with training sets having more distorted images receiving lower human success rates. Many human users had difficulty in distinguishing between human and cartoon faces, especially once distortions were applied. Facial feature occlusions were most problematic; the black stripes placed across some images greatly hindered recognition. As more bars were placed, covering a larger area of the CAPTCHA, people were less likely to correctly identify the image. Based on this finding, life-like cartoon images such as those in Fig. 4.6 were removed from the source image database used in generating the final optimized CAPTCHAs.

Figure 4.6: Samples of embedded images that are removed after the training process [157].

Adversaries were able to solve 0.2% of training CAPTCHAs. Just as with human users, there was an inverse relationship between the intensity of applied distortions and success rates; lightly distorted CAPTCHAs were more likely to be solved than heavily distorted images. The training results revealed that several distortion types, including blurring, dilation, erosion, and warping effects, reduced automated attack rates by less than 50%. As a result, these less effective distortions were not used in generating the final optimized images.

Human performance evaluation

A total of 9,948 attempts were recorded by 1,025 human users on the final optimized FaceD-CAPTCHA images. Results were markedly improved compared to the training phase, with an overall human success rate of 86.6%. This nearly 7-in-8 success rate compares favorably to the 75% and 80% success rates of major commercial CAPTCHAs, such as reCAPTCHA and MSN [22]. Table 4.1 shows results by distortion type.

Table 4.1: Final FaceDCAPTCHA success rates by distortion type

Distortions Used	**Human Success**	**Attack Success**
Blending, noise, rotation	90.3%	0.0%
Blending, noise, rotation, strikeouts	86.6%	0.0%
Blending, noise, rotation, strikeouts, stripes	84.3%	0.0%
Blending, noise, rotation, stripes	86.0%	0.0%
Noise, rotation	97.6%	0.0%
Noise, rotation, strikeouts	87.8%	0.0%
Noise, rotation, strikeouts, stripes	85.8%	0.0%
Noise, rotation, stripes	86.9%	0.0%
Overall	**86.6%**	**0.0%**

One major source of improved human performance in the final images was a reduction in the amount of background blending applied. In the training set, where embedded images were more heavily blended into their background, it was frequently challenging for users to distinguish between genuine and fake facial features. When blending is reduced, as in the final optimized image set, subtle differences are more obvious to human users. Limiting the use of facial feature occlusions, and making occlusions more transparent when they were used, also provided humans with more data to use in distinguishing between image types.

Automated attack evaluation

Picasa and the commercial off-the-shelf face detector were unable to locate any embedded genuine human faces among the final optimized CAPTCHAs. Viola-Jones detected one face each in two CAPTCHAs, but false faces were also detected so its attempts at solving the CAPTCHAs were not successful.

FaceDCAPTCHA was also tested using Zhu's [98] automated attack approach which has been used against the IMAGINATION and ARTiFACIAL CAPTCHAs with 74% and 18% success rates, respectively. The same approach was unable to solve any of the final FaceDCAPTCHA images.

Brute force attacks based on randomly guessing the locations of embedded human faces are similarly unlikely to succeed. Each FaceDCAPTCHA image contains two to four genuine human faces. Each face is approximately 80×80 pixels in size, with an overall CAPTCHA image size of 400×300 pixels. Thus, the average chance of a single brute force attempt at correctly solving FaceDCAPTCHA is slight:

$$\left(\frac{1}{3}\right) \sum_{i=2}^{4} \left(\prod_{j=1}^{i} \frac{(80 \times 80)j}{(400 \times 300) - (i - j)} \right) = 0.2264\% \tag{4.7}$$

Since a new image is chosen at random from a large set each time FaceDCAPTCHA is presented, a would-be attacker would likely have to wait hundreds of times before seeing the same image again if their attempt is unsuccessful. This dramatically slows the process of attacking FaceDCAPTCHA. In testing with a random guess-based breaking algorithm on an Intel Core 2 Duo 2.2GHz processor, none of the 500 tested CAPTCHAs were successfully broken after over 15 hours of execution.

4.3 Summary

Through its use of a gradient descent-based training-testing process, FaceDCAPTCHA generates face detected-based CAPTCHA images with distortions optimized to improve the CAPTCHA's performance. Compared to its predecessor, Face Detection CAPTCHA, FaceDCAPTCHA has significantly improved human success rates (from 71.8% to 86.6%) and also nearly eliminated the likelihood of successful automated attacks. These improvements do come at a cost, however, with significant training, both manual and automatic, being required to generate these optimized images.

Chapter 5

fgCAPTCHA

5.1 Proposed approach

5.1.1 CAPTCHA design

fgCAPTCHA [163] builds upon the improved performance of FaceDCAPTCHA by fully automating the process of generating face detection-based CAPTCHA images. As with its predecessors, fgCAPTCHA requires users to select, without error, all genuine human face photographs embedded in a visually distorted composite image. Cartoons, sketches, and photographs of non-human animals are also present to serve as fake distractor images. Fig. 5.1 shows an example of a correctly solved fgCAPTCHA test.

While FaceDCAPTCHA's gradient descent optimization process provides significantly improved human success and automated attack success rates compared to the original Face Detection CAPTCHA, its use of "human in the loop" training to optimize the applied distortions is impractical. It is difficult to find a sufficiently large number of human participants to provide optimization training data in a closed setting. Public use of unoptimized training images is undesirable since these images are less likely to be readily solvable by humans and more likely to be vulnerable to automated attack. fgCAPTCHA addresses these shortcomings by using a genetic learning algorithm to fully automate the distortion optimization

Figure 5.1: Sample fgCAPTCHA image with correct answers circled in red [163].

process; only images which have been optimized to provide high human success rates and ensure low automated attack rates are publicly used. fgCAPTCHA also improves upon FaceDCAPTCHA's process by optimizing applied distortions on a per-image basis, rather than on a source image set basis. This yields finer-grained settings along with the potential for further improved human success rates.

5.1.2 Generation process

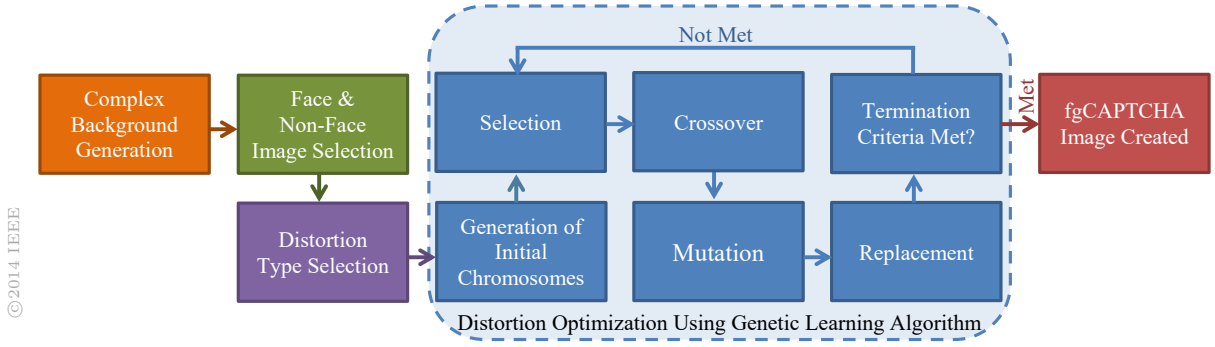


Figure 5.2: Steps involved in generating fgCAPTCHA images [163]. A complex background is first generated and face and non-face images are selected to be embedded. A genetic learning algorithm then begins an optimization process to generate and evaluate test CAPTCHAs to determine the distortion settings that yield the best performing CAPTCHAs. These settings are then used to generate the final fgCAPTCHA image.

Generation of fgCAPTCHA images is a multi-step process as shown in Fig. 5.2. It involves several distinct phases: complex background generation, genuine face and fake non-face image selection, distortion type selection, and genetic learning algorithm-based distortion optimization. The generation process can be represented as:

$$C = F(n_{min}, n_{max}, \phi, I_{genuine}, I_{fake}) \tag{5.1}$$

where function F creates a new CAPTCHA containing a total of between n_{min} and n_{max} embedded images taken from sets $I_{genuine}$ and I_{fake}. Distortion settings (distortion types and distortion intensities) selected from ϕ are applied to the rendered composite, yielding CAPTCHA C. The goal of the CAPTCHA generation process is to find distortion settings that maximize the chance that humans will be able to solve the CAPTCHA while minimizing the likelihood of successful automated attacks by computer algorithms. This can be shown as:

$$\text{argmax}_{\varphi}\, P(C_{\varphi}) = P_H(C_{\varphi}) - P_A(C_{\varphi}) \tag{5.2}$$

where C_{φ} is a CAPTCHA with distortion settings φ applied, P_H is the likelihood humans can solve the CAPTCHA, P_A is the likelihood automated attacks can solve the CAPTCHA, and P is the difference between the two likelihoods.

Without including humans in the loop during the CAPTCHA generation process, it is impossible to know the actual values of P_H and P for a given CAPTCHA. Instead, a simulation process is used to model human performance. The results of the simulation are used to calculate a fitness value for each generated CAPTCHA:

$$V\left(C_{\varphi}\right) = S_H\left(C_{\varphi}\right) - S_A\left(C_{\varphi}\right) \tag{5.3}$$

where S_H is the simulated likelihood of human success, S_A is the likelihood of a successful automated attack, and fitness value V is the difference between the two likelihoods. Higher

values of V indicate a CAPTCHA where it should be relatively easier for humans to detect the faces while being more difficult for automated attackers to successfully complete the face detection task. These attacks can be modeled by performing automated face detection and comparing detected face locations against known face locations. fgCAPTCHA uses the Viola-Jones algorithm [153] to locate embedded faces. The face locations indicated by the algorithm are compared against the actual embedded genuine human face locations, with the automated attack rate being the percentage of genuine human faces that are found. Lower values resulting from Eq. 5.4 are better:

$$S_A\left(C_\varphi\right) = \frac{d_{correct} - d_{false}}{n} \leq 1.0 \tag{5.4}$$

where n is the number of embedded human faces in CAPTCHA image C with distortion φ applied, $d_{correct}$ is the number of human faces correctly detected by the algorithm, and d_{false} is the number of false human face detections.

Since there is no direct way of simulating human performance, fgCAPTCHA indirectly models human success rates using image quality metrics. The Structural Similarity (SSIM) model, a metric designed to mimic the human visual system, compares distorted and undistorted versions of embedded images to look for differences in linear correlation, luminance, and contrast [164]. To model human performance, SSIM is performed on each embedded image. Values closer to 1.0 signify that the tested images are more similar and, hopefully, the distorted version will then be relatively easier for humans to solve. The modeled human success rate is an average of all SSIM values:

$$S_H\left(C_\varphi\right) = \frac{\sum_{j=0}^{n} SSIM\left(C_{j\varphi}\right)}{n} \leq 1.0 \tag{5.5}$$

where n is the number of genuine human faces embedded in CAPTCHA image C and $SSIM\left(C_{j\varphi}\right)$ is the resulting SSIM value when distortion settings φ are applied to the embedded image j.

Background generation

The creation of a new fgCAPTCHA image begins with the generation of a 400×300 pixel background composed of many overlapping rectangles in various colors and sizes. The individual colored rectangles have their colors chosen at random from a list of 56 common hues, including skin tones. Height and width are based on a fraction of the overall image size, randomly scaled, so that:

$$s = \left\{ \frac{r}{10} \min(height, width) \middle| 0.75 \leq r \leq 1.25 \right\} \tag{5.6}$$

where s is the resulting size in pixels for one side of the rectangle, $height$ and $width$ are the overall height and width of the background, and r is a random real-valued scaling factor. Colored rectangles are scattered across the entire background until at least 95% is covered. This provides a complex pattern to interfere with the rectangular features used by the Viola-Jones detector and other similar face detection algorithms. The random sizes and colors make it difficult to isolate embedded images, and in some cases, lead algorithms to falsely detect faces in the background.

Image selection

Once the background is generated, a total of four to five genuine face and fake non-face distractor images are selected to be embedded:

$$n_{total} = \left\{ n_{genuine} + n_{fake} \;\middle|\; n_{genuine} \geq 2, n_{fake} \geq 1, n_{total} = \{4, 5\} \right\} \tag{5.7}$$

Here $n_{genuine}$, n_{fake}, and n_{total} represent the number of embedded genuine faces, fake non-faces, and total images, respectively. At least two genuine face images are present to prevent a single guess from successfully solving the CAPTCHA and at least one image is a non-face image to provide a false target in case attackers can detect the location of embedded images. Each embedded image is scaled to approximately 100×100 pixels prior to placement.

This size corresponds with the area covered by a fingertip for accurate use on touchscreen devices. The images are placed at randomly selected coordinates within the background, ensuring that the images do not overlap with each other or the outside boundary of the CAPTCHA. An example of a preliminary CAPTCHA with embedded images appears in Fig. 5.3.

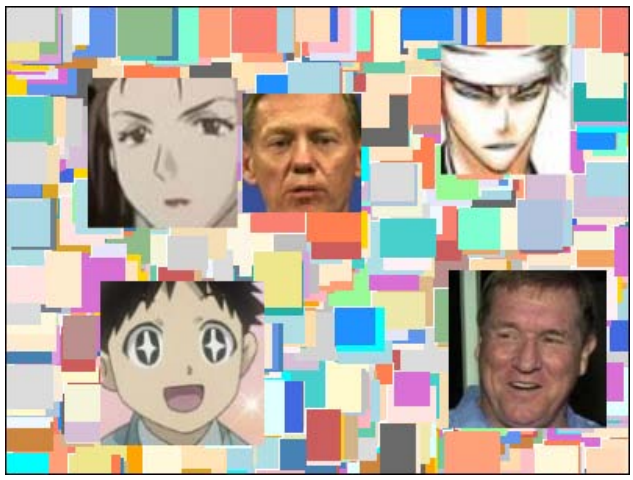


Figure 5.3: Example of a new undistorted fgCAPTCHA [163].

Distortion selection

The distortions applied to each CAPTCHA have a significant impact on human and automated attack success rates. During design, 10 distortion types were identified that yield the best performance. Each distortion type has a range of possible intensities adjusted by the various parameters shown in Table 5.1.

In this step, the various distortion types are compared to find the types that provide the best performance when applied to the intended CAPTCHA. Each distortion type is applied at eight different intensities evenly spread over its range. Performance or fitness values are calculated for each of the resulting images using Eq. 5.3. The results are ranked by their fitness, with those distortion types yielding the top 50% most-fit CAPTCHAs selected for further use. A Cartesian product is created combining two each of the best-fit distor-

Table 5.1: fgCAPTCHA distortion types [163]

Distortion Type	**Class**	**Applied**	**Parameters Adjusted**	**Intensity Range**
Erosion	Degradation	Globally	Pixel radius	3.0-3.5
Lighten	Degradation	Globally	Scale histogram range	0.30-0.45
Resolution modification	Degradation	Locally	Scale factor	1:3.5-1:5
Width scaling	Geometric	Locally	Scale factor	1.5-4.0
Height scaling	Geometric	Locally	Scale factor	1.5-3.0
Rotation	Geometric	Locally	Degrees of rotation	60-180
Piecewise scaling	Geometric	Locally	Scale factor	1.5-3.5
Periodic noise	Noise	Globally	% of image removed	5-7.5
Salt-and-pepper noise	Noise	Globally	% of pixels to replace	10-20
Speckle noise	Noise	Locally	Variance	2-5

tion types. Experience has shown that applying two distortion types to each CAPTCHA represents a good balance between making images too simple for automated attacks (one distortion type) or too hard for human users (three or more distortion types). Some distortion type pairs known to perform poorly are discarded, with the rest continued to the next distortion optimization step.

The distortion types can be classified into three categories: geometric, noise-based, and degradation distortions. Geometric distortions alter the shape, size, or position of embedded images. Width and height scaling change the dimensions of an image. Piecewise scaling leaves the overall dimensions of the image untouched but changes the relative proportions of sections of the image. For example, the left half of an image might be compressed to take up 50% less space than before while the right half is stretched to fill the available space. The rotation distortion rotates the image around a center axis. Any portion of the image

falling outside the original dimensions is removed. Rotation can be performed using the transformation matrix [158]:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos(\varphi) & -\sin(\varphi) \\ \sin(\varphi) & \cos(\varphi) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{5.8}$$

Here, (x, y) are the original coordinates of a pixel, φ is the degree of rotation to be applied in radians, and (x', y') are the adjusted coordinates of the pixel.

Noise-based distortions add interference not present in the original image. Salt-and-pepper noise changes the values of the specified percentage of pixels to the maximum or minimum possible value, thus having the effect of adding randomly discolored pixels to the overall CAPTCHA. Speckle noise modifies the values of individual pixels in a pattern that is uniformly distributed with a mean of 0 and a variance specified by the distortion intensity. Periodic noise creates a repeating pattern of darkened bars across the entire image. It can be generated by [165]:

$$v_d(x, y) = \max\left(0, \min\left(\frac{v(x, y) + (\sin(\frac{y+1}{\varphi}) * 255)}{2}, 255\right)\right) \tag{5.9}$$

where $v(x, y)$ is the original pixel value, a number between 0 and 255, at coordinates (x, y). $v_d(x, y)$ is the distorted pixel value and φ is the distortion intensity.

Degradation distortions are designed to reduce detail or contrast, making it difficult to find the embedded images. The lighten distortion increases the luminance of each pixel by a specified percentage, effectively reducing the contrast of a CAPTCHA. Erosion works on the entire CAPTCHA in successive blocks. It compares the values of each pixel with those of its neighbors and eliminates unique values, reducing fine detail.

Resolution modification is performed as a pair of bilinear resizing operations, the first reducing the size of the image and the second expanding it to its original size. As pixel data

is lost, a blocky-looking image emerges. The bilinear resizing operation can be represented using [159]:

$$v(x', y') = ax' + by' + cx'y' + d \tag{5.10}$$

where $v(x', y')$ is the pixel value of coordinates (x', y') and coefficients a, b, c, d can be solved using four equations in four unknowns for the four neighbors of (x', y').

Distortion optimization

Once the distortion type pairs have been determined, optimal intensities for each distortion must be found. This is a complex problem with a huge search space; therefore, brute force exploration is not feasible. fgCAPTCHA instead uses a genetic learning algorithm (GA) to efficiently identify optimal distortion settings. Genetic algorithms are modeled on the biological process of evolution [166]. They work by producing successive generations of candidate solutions, referred to as chromosomes, to find the distortion settings that generate the optimized CAPTCHA. The algorithm includes several steps (input parameters are summarized in Table 5.2) as described below.

Step 1: Generate Initial Chromosomes - The algorithm begins by generating an initial set of 150 chromosomes, each representing one possible combination of distortion settings. The chromosomes contain two genes, each encoding a single distortion type and its associated real-valued intensity. Distortion types are selected from the list of approved distortion type pairs and their intensities are randomly set to a value within the distortion type's specified range. After the chromosomes are generated, a fitness value is calculated for each using Eq. 5.3. Since each distortion type has a distinct range of intensities, genetic algorithm operations such as crossover must be performed only between chromosomes with the same distortion types. Thus, the chromosomes are organized into groups based on their distortion types as shown in Fig. 5.4. To ensure genetic diversity within each group, a minimum of two chromosomes per group is maintained.

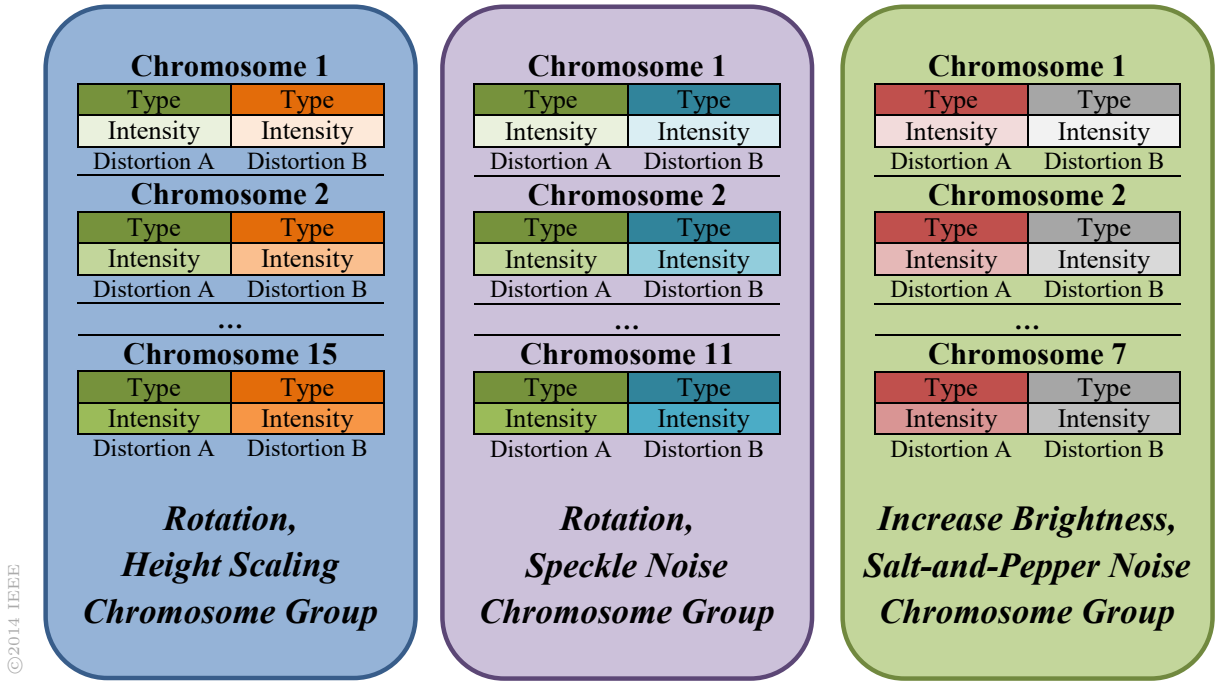


Figure 5.4: Example of chromosome groups [163].

Step 2: Select Candidates for Next Generation - A roulette wheel-based process is used to select chromosomes to create the next generation. The process selects chromosomes at a rate proportional to their fitness [167]:

$$p_i = \frac{\alpha_i}{\sum_{i=0}^{n} \alpha_i} \tag{5.11}$$

where n is the total number of chromosomes, α_i is the fitness of chromosome i, and p_i is the probability chromosome i will be selected. Roulette wheel selection works by first summing the fitness values of all chromosomes, yielding T. Then, for each chromosome, a random value λ, between 0 and T, is selected. The list of chromosomes is iterated through, adding their fitness values, until the sum is greater than or equal to λ. The chromosome whose fitness value brings the sum over λ is selected to create the next generation [167].

Step 3: Perform Crossover - In the crossover step, the values from two parent chromosomes are used to produce two child chromosomes. Approximately 80% of parent chro-

mosomes are randomly selected to participate in this process. A variation on single-point crossover, shown in Fig. 5.5, is used to accommodate the real-valued distortion intensities stored in the genes. As with single-point crossover, prior to the crossover point, child chromosomes $K1, K2$ inherit directly from their parent chromosomes $J1, J2$ so that for a given gene x on each chromosome, $K1_x = J1_x$ and $K2_x = J2_x$. After the crossover point, a weighted combination of the two parent chromosomes is used to simulate the value changes that would occur with a binary string representation in traditional single-point crossover. Here, $K1_x = \frac{1}{4}J1_x + \frac{3}{4}J2_x, K2_x = \frac{1}{4}J2_x + \frac{3}{4}J1_x$ for gene x.

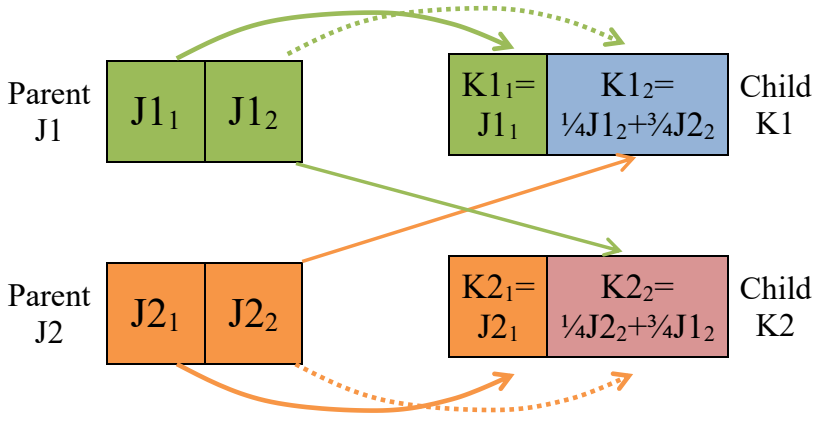


Figure 5.5: Demonstration of crossover process between parent chromosomes $J1$ and $J2$ to create child chromosomes $K1$ and $K2$ [163]. The crossover point is between genes 1 and 2.

Step 4: Conduct Mutation - To prevent stagnation of results at local optima, mutation is applied to approximately 5% of gene values. This helps to ensure the entire solution space is searched rather than just values near those of the parent chromosomes. The traditional mutation approach of randomly flipping bits in a binary-encoded gene value does not work with real-valued genes. Instead, the existing gene value is averaged with a new random value when mutation is performed:

$$m = \left\{ \frac{c+n}{2} \middle| dist_{min} \leq n \leq dist_{max} \right\} \tag{5.12}$$

where c is the existing value of the gene, n is a random real-valued number between the $dist_{min}$ and $dist_{max}$ minimum and maximum intensity values allowed for the distortion, and m is the mutated gene value.

Step 5: Run Replacement - Once a new generation of chromosomes has been created, an $\lambda + \mu$-update replacement process is used to select the chromosomes that will be retained. This method keeps the chromosomes with the best fitness values from both the parent and child generations, thereby preserving good chromosomes from the parent generation that might otherwise be lost with a traditional generational replacement.

Step 6: Evaluate Termination Criteria - Once replacement has occurred, the fitness values for all chromosomes are compared. The best fitness value is recorded for each generation. The genetic learning algorithm can terminate if enough generations have been run or if the best fitness value stagnates. Otherwise, operation of the genetic learning algorithm continues and the chromosomes resulting from the replacement process are provided as input to the selection step to create a new generation. Actions are determined by:

$$\text{Action} = \begin{cases} \text{Complete} & \text{if } g \geq 100 \\ \text{Complete} & \text{if } g \geq 50 \text{ and } best_g \leq 1.01 * best_{g-5} \\ \text{Continue} & \text{otherwise} \end{cases} \tag{5.13}$$

Here, g is the number of the current generation and $best_g$ is the best fitness value for generation g.

Step 7: Completion - The genetic learning process stops once the termination criteria have been satisfied. To ensure that any readily attackable images do not see public use, any CAPTCHAs with computer-based attack success rates of $S_A = 1.0$ are discarded. The remaining CAPTCHAs with the best fitness values are recorded along with their embedded image coordinates so they can be used as tests. Examples of generated CAPTCHAs presented to users are shown in Fig. 5.6.

Table 5.2: fgCAPTCHA genetic algorithm details [163]

Parameter	Type or Value
Initial Chromosomes	150
Selection Method	Roulette Wheel
Crossover Rate	80%
Crossover Method	Modified Single-Point
Mutation Rate	5%
Replacement Method	$\lambda + \mu$-update
Termination Criteria	Stagnation or Maximum Generations
Stagnation Condition	Minimum 50 generations, then less than 1% change in best fitness over 5 generations
Maximum Generations	100

5.2 Experimental results and analysis

This section provides the details of image databases used, participants, and protocol followed for designing and evaluating the performance of fgCAPTCHA along with the results and analysis.

5.2.1 Image databases

fgCAPTCHA uses publicly available photographs from the LFW face database [161] for the genuine human face images. Cartoons and high-quality sketches from Photobucket [162] comprise the fake non-face images used in the CAPTCHA.

5.2.2 Participants and testing protocol

Evaluation of fgCAPTCHA was conducted with the help of 2,600 participants. Users were unsupervised and asked to access a website protected by fgCAPTCHA using a browser and computing device of their choice. For each attempt, one CAPTCHA was presented at a time. Users were asked to continue solving CAPTCHAs until they were successful.

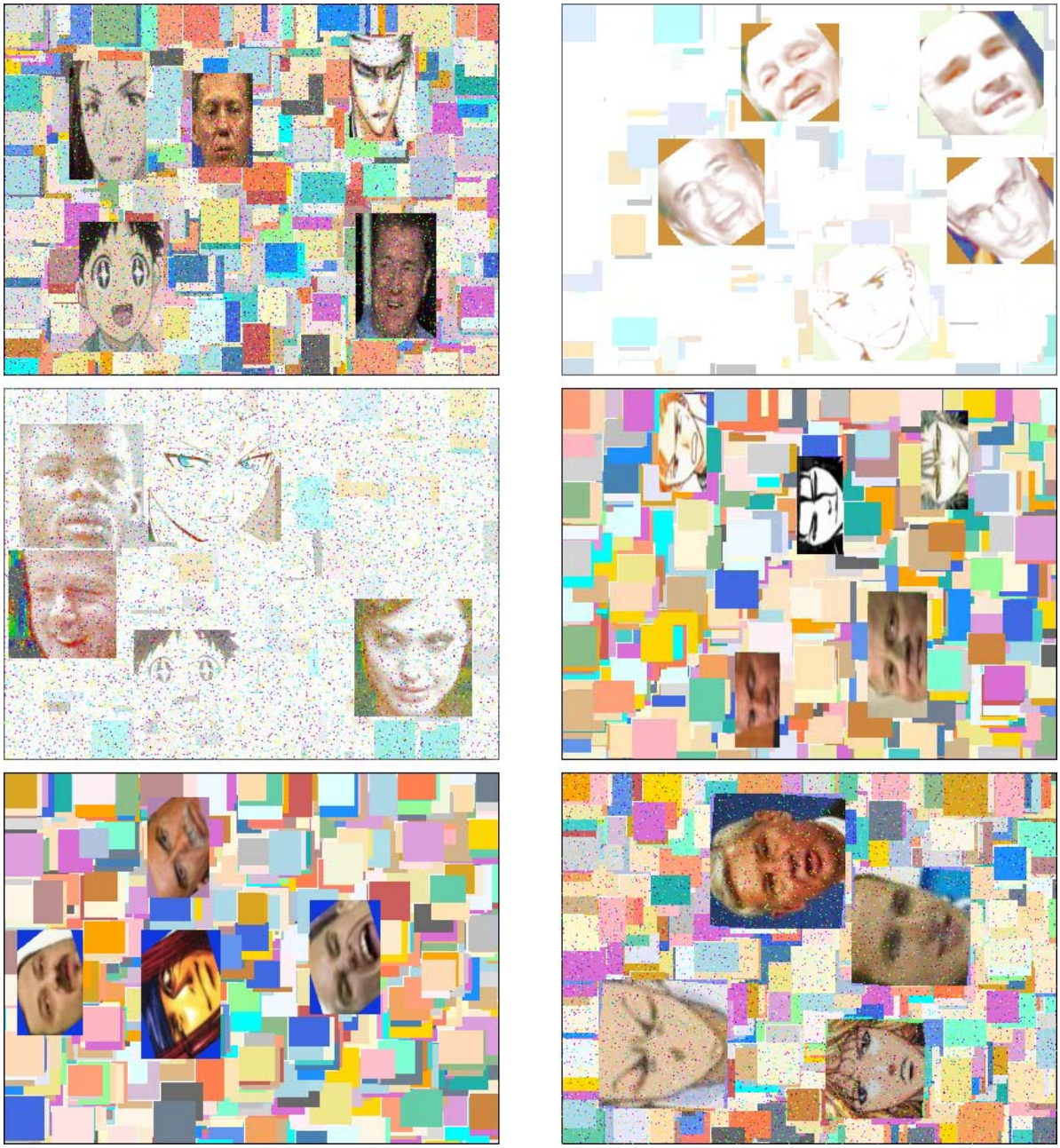


Figure 5.6: Samples of generated fgCAPTCHA images [163].

An additional 17 users participated in detailed mobile device testing using a combination of tablets and smartphones. These users compared fgCAPTCHA with two other popular CAPTCHAs, namely text-based reCAPTCHA [81] and image-based IMAGINATION [97].

Their success rates were recorded, and the users also provided a ranking of the CAPTCHAs by their ease of use.

Automated attack testing was performed using the Viola-Jones face detection algorithm [153] and two commercial off-the-shelf (COTS) face detection packages. The faces detected by software were compared to the actual face locations. If any portion of the detected face overlapped an actual face, the face was considered to be correctly found. An automated attack was considered successful if all human faces in a CAPTCHA were found without any false detections.

5.2.3 Analysis

Human performance evaluation

To collect data and evaluate the effectiveness of fgCAPTCHA, over 40,000 attempts by over 2,600 users were recorded on the set of 750 rendered CAPTCHAs. The human success of solving fgCAPTCHA is dependent on the complexity and level of distortions applied using genetic learning. With simple distortions such as rotation and height scaling, the human success rate was 97%; but complex distortions such as resolution modification and adding noise affect the performance significantly. In the experiments, average human performance across all variations was 87.9%. Detailed results are summarized in Table 5.3. From these results, we can infer that, in general, geometric distortions such as height scaling and rotation yield higher human success rates. These distortions do not fundamentally alter the appearance of images; they only resize or reposition facial features, allowing human users to easily detect the embedded faces. Noise-based distortions yield similar performance. Degradation distortions result in lower accuracies as, in some cases, they destroy the fine details needed to distinguish images. This effect is pronounced when sketches are used for non-face images. When degradation distortions are used, humans have significant difficulty in distinguishing between human face photographs and non-real face sketches.

Table 5.3: fgCAPTCHA success rates for distortion type pairs [163]

Distortions Used	**Human Success**	**Attack Success**
Rotation, height scaling	97.0%	0.0%
Lighten, salt-and-pepper noise	92.9%	0.0%
Erosion, salt-and-pepper noise	92.6%	0.0%
Rotation, speckle noise	91.7%	0.0%
Height scaling, width scaling	91.4%	0.0%
Rotation, width scaling	89.9%	0.0%
Piecewise scaling, width scaling	88.7%	0.0%
Rotation, salt-and-pepper noise	88.5%	0.0%
Lighten, speckle noise	88.2%	0.0%
Lighten, rotation	88.1%	0.0%
Piecewise scaling, salt-and-pepper noise	87.6%	0.0%
Salt-and-pepper noise, height scaling	87.3%	0.0%
Piecewise scaling, height scaling	85.8%	0.0%
Salt-and-pepper noise, width scaling	85.8%	0.0%
Resolution modification, height scaling	85.1%	0.0%
Lighten, resolution modification	83.3%	0.0%
Salt-and-pepper noise, speckle noise	83.3%	0.0%
Periodic noise, width scaling	81.3%	0.0%
Lighten, width scaling	81.1%	0.0%
Erosion, resolution modification	81.0%	0.0%
Lighten, height scaling	81.0%	0.0%
Resolution modification, salt-and-pepper noise	80.0%	0.0%
Overall	**87.9%**	**0.0%**

Additionally, 17 participants evaluated fgCAPTCHA along with two other popular CAPTCHAs, reCAPTCHA and IMAGINATION, on mobile devices using a combination of tablets and smartphones. In this evaluation, fgCAPTCHA achieved the best mobile device human success rate with 88.2% accuracy. Seventy percent of test participants indicated that fg-

CAPTCHA was easiest to use, with several individuals commenting that it could be completed quicker than other CAPTCHAs. Participants specifically appreciated the touch-friendly nature of fgCAPTCHA that allowed it to be solved with just a few taps to the screen. Fig. 5.7 illustrates these comparisons along with automated attack success rates.

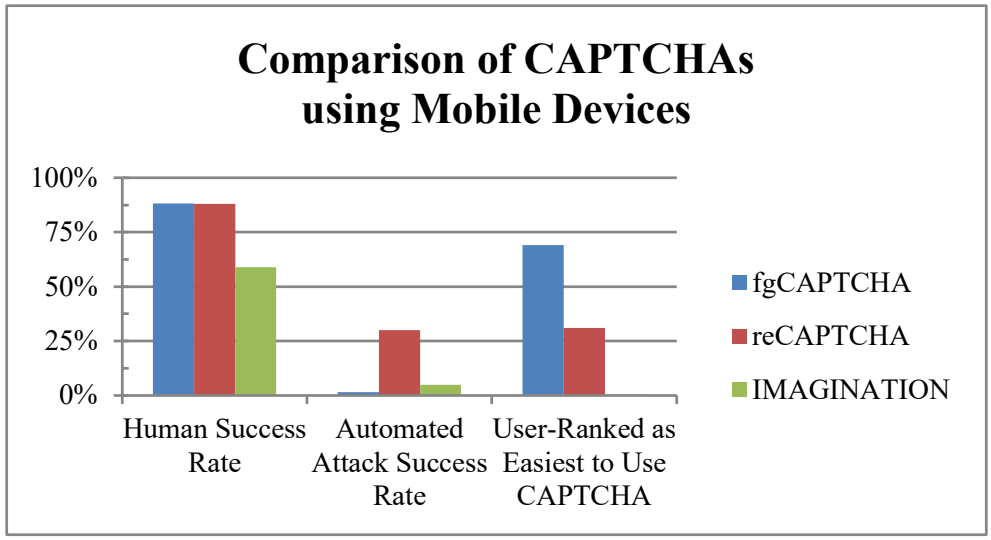


Figure 5.7: Comparison of CAPTCHAs [163] when tested by humans on mobile devices, contrasted with automated attack success rates [56, 98].

Automated attack evaluation

In the automated attack evaluation, none of the three face detection algorithms were able to correctly solve any of the tested CAPTCHA images. In cases where the algorithms were able to detect some human face images, other faces were either missed or falsely detected. This is expected since the widely-used Viola-Jones face detector is incorporated into the CAPTCHA generation process and CAPTCHAs where the Viola-Jones detector locates all faces are automatically discarded.

It is unlikely that an automated brute force attack on fgCAPTCHA would be successful. Each CAPTCHA contains two to four human face images, each being approximately 100×100

pixels in size. Including the one-in-three chance of guessing the number of embedded images, the likelihood of one random guess at solving the CAPTCHA being accurate is approximately,

$$\left(\frac{1}{3}\right)\sum_{i=2}^{4}\left(\prod_{j=1}^{i}\frac{(100\times 100)j}{(400\times 300)-(i-j)}\right)=0.6173\% \tag{5.14}$$

Since new CAPTCHA images are presented on each attempt, attackers are unable to use previous guesses to improve the accuracy of future attempts. Attackers must make a new random guess each time. Thus, the effective attack success rate is 6-in-1000, thereby significantly enhancing security of the online environment using fgCAPTCHA.

5.3 Summary

fgCAPTCHA improves on its predecessors, FaceDCAPTCHA and Face Detection CAPTCHA, in a number of ways:

1. By incorporating improved visual distortions that further strengthen the security of the CAPTCHA without sacrificing human ability to solve the tests.

2. By using color images and a genetic learning algorithm-based generation process that increases human success rates while also reducing the automated attack success rates.

3. By removing the dependency on humans for distortion parameter selection and optimization, making the CAPTCHA generation process highly scalable.

4. By accounting for design requirements tailored to the devices commonly used to view the CAPTCHA. Screen size and resolution can vary significantly even among devices of the same type. A well-designed CAPTCHA must work effectively across the entire spectrum of computing devices, from smartphones, where it may be the only item on-screen, to tablets and computers, where it is part of a larger webpage.

This combination of improvements provides a layer of security that is both effective and user-friendly. fgCAPTCHA works efficiently on both the touchscreens used by tablets and smartphones and on traditional computers, achieving a high 88% human success rate during evaluation while maintaining a near 0% automated attack rate. This combination of low attack rates, high human success rates, and convenient mobile device usage offers significant benefits over CAPTCHAs in widespread use today.

Chapter 6

FR-CAPTCHA

6.1 Proposed approach

6.1.1 CAPTCHA design

While the face detection task used in fgCAPTCHA and the other previously described CAPTCHAs offers a good balance between ease-of-use for legitimate human users and resiliency against automated attack, increasingly sophisticated face detection algorithms and image processing techniques may eventually solve these CAPTCHAs. To address this future risk, *FR-CAPTCHA* [168, 169] instead uses tests based on face recognition rather than face detection. Face recognition is a problem that the human mind solves every day, usually without conscious effort. Humans can recognize images with differing poses, illumination, and facial expression like those shown in Fig. 6.1 as being of the same person [170], but automated face recognition algorithms struggle with this task, especially when visual distortions are added to the images [171]. This disparity in performance between humans and automated algorithms [172] when viewing visually distorted images makes face recognition an ideal test for use in a CAPTCHA.

FR-CAPTCHA presents users with a composite color image containing visually distorted photographs of human faces as well as cartoons, sketches, and photographs of other objects.

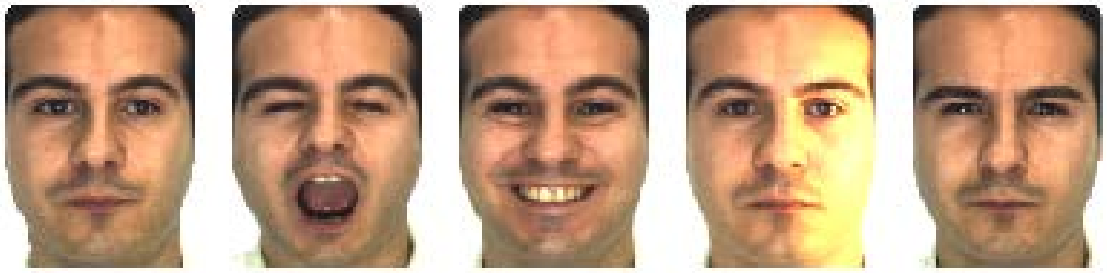


Figure 6.1: Humans can easily recognize faces with different natural variations in pose, expression, and illumination [169]. Automated face recognition algorithms struggle with this task.

The embedded images include two pairs of photos where each pair shows the same person in two different poses. To solve the CAPTCHA, users must select both of the photos from the same pair (i.e., both photos of the same person) without any mis-selections. Fig. 6.2 shows an example of a solved FR-CAPTCHA.

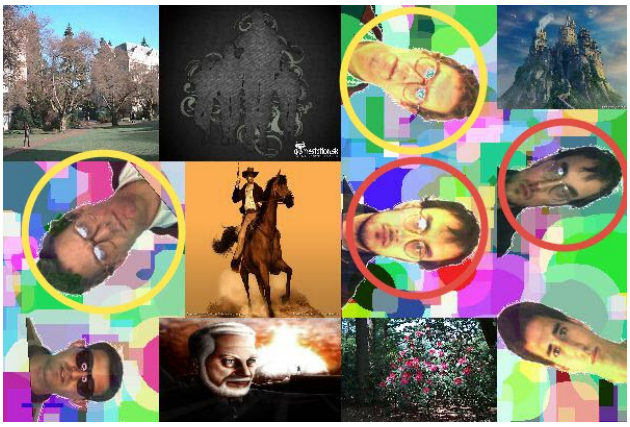


Figure 6.2: Sample FR-CAPTCHA image with correct answer pairs circled in red and yellow.

6.1.2 Generation process

The FR-CAPTCHA generation process can be represented as:

$$C = F(\phi, I_{face}, I_{nonface}) \tag{6.1}$$

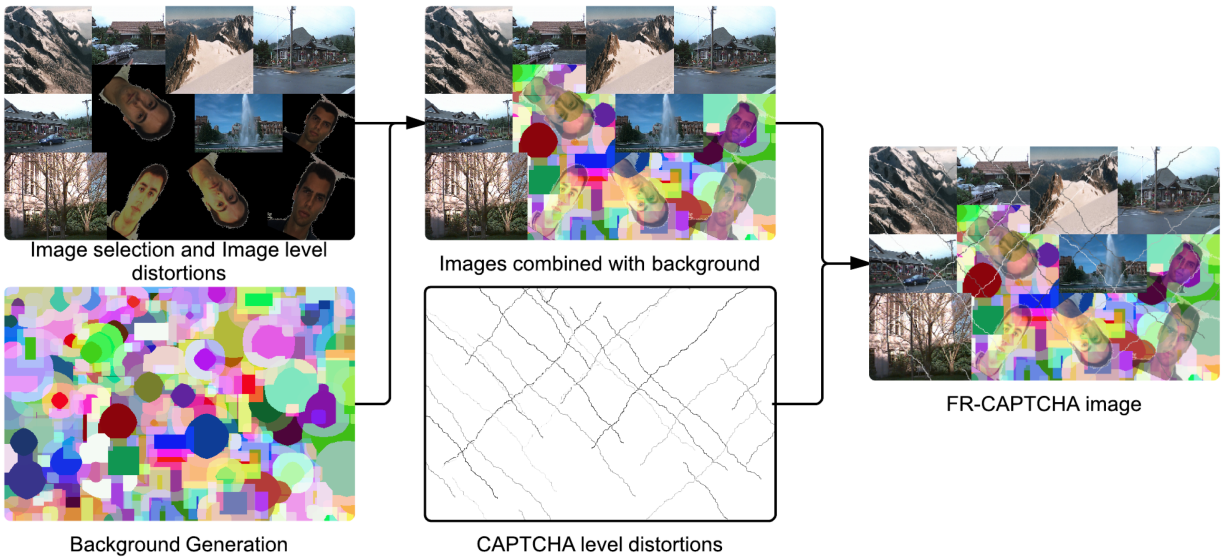


Figure 6.3: Steps involved in generating FR-CAPTCHA images [169]. An image is created using a randomly selected set of human faces and other non-face images after applying different amounts of rotation to each image. A background of the same size is used generating various colored shapes. The background and image are blended together, then further processed to add noise, illumination variance, and false edges. The resulting image is a FR-CAPTCHA.

where function F represents a series of image processing operations applied using a set of generation parameters, ϕ, and source images, I_{face} and $I_{nonface}$, to generate the FR-CAPTCHA image C. Like FaceDCAPTCHA, FR-CAPTCHA uses an adversarial learning approach along with gradient descent optimization to learn the optimal set of parameters that maximize the human success rate at solving the CAPTCHA while minimizing the automated attack success rate. The FR-CAPTCHA generation process can be divided into two phases: (1) creation of and performance collection using training CAPTCHAs, and (2) generation of the final FR-CAPTCHA images using optimized parameters.

CAPTCHA generation parameters

The parameters specified in ϕ control three aspects of generated FR-CAPTCHA images:

- The parameter set $\mathbf{B}$ specifies weights to use when blending objects in the CAPTCHA. The weights control the difficulty in distinguishing between blended foreground and background objects.
- The sizes and colors of the shapes that comprise the background image are determined by parameter set $\mathbf{S}$.
- The parameter set $\mathbf{D}$ specifies settings for several distortion operations that are applied to CAPTCHA images. The distortion types available include rotation and added noise.

FR-CAPTCHA's gradient descent optimization process uses results from people and automated attack adversaries attempting to solve training CAPTCHAs to find optimal values for the ϕ generation parameters. The initial parameter values used in generating the training CAPTCHAs impact the effectiveness of the optimization process; poorly chosen values for the training images can yield suboptimal results. To ensure the best possible results, the training CAPTCHAs' generation parameters were determined using small scale experiments with both humans and face recognition algorithms. The values used are shown in Table 6.1.

Both training and final optimized FR-CAPTCHA images are generated using the multi-step process shown in Fig. 6.3 and described below:

Image selection

Creation of an FR-CAPTCHA image begins by selecting n_{face} face and $n_{nonface}$ non-face images from sets I_{face} and $I_{nonface}$, respectively, so that:

$$n_{total} = \left\{ n_{face} + n_{nonface} \;\middle|\; n_{face} = \{4, 5, 6\}, n_{total} = 12 \right\} \tag{6.2}$$

Table 6.1: Parameter values used in generating FR-CAPTCHA training images [169]

Parameter	**Value**
No rotation	0°
Low rotation	0° - 60°
Medium rotation	30° - 120°
High rotation	45° - 170°
No blend	100% foreground
Low blend	80% foreground
Medium blend	65% foreground
High blend	50% foreground
No global distortions	illumination = false, false edges = false
Low global distortions	either illumination or false edges
High global distortions	illumination = true, false edges = true

To allow for FR-CAPTCHA's face recognition test, the face images must be selected so that two pairs of photographs are present, where each pair includes two photographs of the same person.

Image placement

After their selection, each image to be embedded is randomly resized to between 100×125 and 175×150 pixels. Each image is also rotated by a stochastically chosen angle within the range $[D_{Rmin}, D_{Rmax}]$.

Each embedded image is placed at a randomly selected location within the boundaries of the planned 600×400-pixel CAPTCHA. If an image being placed overlaps with a previously placed image, the images' pixels are blended together using weighted average blending. The weighting varies depending on whether the overlapping image is a face; face images are assigned higher weights for the blending process than non-face images to facilitate human face recognition. These weights are stored in the parameter B_E.

Background generation and blending

CAPTCHA generation continues with the creation of a 600 × 400-pixel background image. A series of circles, rectangles, crosses, and ellipses in varying sizes and colors are controlled by parameter $\boldsymbol{S}$. Skin-color patches are also included in the background to confuse skin color-based segmentation algorithms. Finally, a series of erosion and dilation operations are performed on the background to eliminate sharp edges present from the placed objects.

Once the erosion and dilation operations are completed, the generated background is blended with the foreground containing the already placed images using a weighted average blending scheme. Weights assigned to the background and the foreground are stored in parameters $\boldsymbol{B_B}$ and $\boldsymbol{B_F}$, respectively. The weight assigned to the foreground is higher in regions where an overlap between the placed images already existed. This ensures that every region in the FR-CAPTCHA image is discernible by humans.

Global distortion application

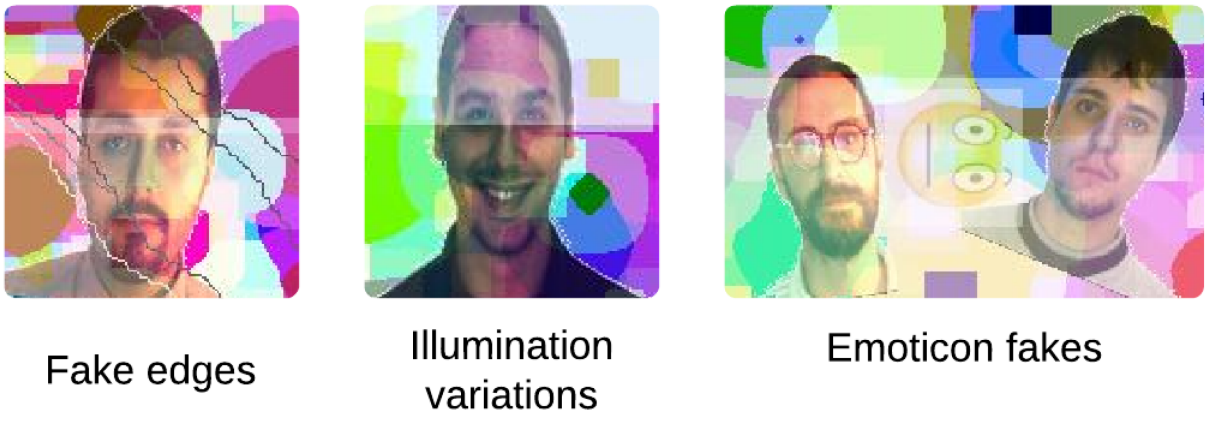


Figure 6.4: Samples of distortions applied during the FR-CAPTCHA generation process [169]. Fake edges are free-form lines drawn throughout the CAPTCHA as in the left image. Illumination variations make portions of the image lighter or darker, such as how the upper half of the face in the middle image is lighter than the bottom half. Emoticon fakes are emoticon-style images inserted into the CAPTCHA, such as the emoticon shown between the two human faces in the right image.

To complicate the process of image segmentation for a would-be attacker, a series of distortions are applied to the blended CAPTCHA image. First, irregular variations in illumination are added by dividing the CAPTCHA into a variable number of non-uniform grids. These grids are then subjected to gamma adjustment with differing gamma values so that some grids are made lighter and others darker.

Next, a random number of irregularly oriented jagged lines are drawn in a free-form style throughout the image to create false edges. Then, emoticon images are blended into the background at randomly selected locations to further confuse a would-be attacker. Finally, a percentage of the total pixels, controlled by parameter D_C, are randomly chosen to have their values corrupted. This introduces noise into the image that can impair the operation of face detection and face recognition algorithms. Fig. 6.4 illustrates the effect of several of these distortions.

CAPTCHA parameter training

Training CAPTCHAs generated using the parameter settings in Table 6.1 are used to learn the useful sets of parameters for generating the final FR-CAPTCHA images. A set of parameters is considered to be useful if, when applied to generated images, humans can successfully solve the resulting CAPTCHAs but the automatic face detection and face recognition adversary algorithms cannot. The useful sets of parameters ϕ_u are learned using:

$$\phi_u = Train_{(C=F(\phi_i, I_{face}, I_{nonface})}(H = 1, Ad = 0) \tag{6.3}$$

where H denotes the human response to solve the CAPTCHA and Ad denotes the response from an automated attack adversary. A correct response is depicted with the value 1; an incorrect response is depicted by 0. Out of the possible parameter sets ϕ_i, only the useful parameter sets ϕ_u are chosen according to the specified constraint on H and Ad. The constraint denotes the ideal criteria that a CAPTCHA has to fulfill: maximum human performance and minimum automated attacker performance. Parameter sets conforming to

other constraints are not useful for FR-CAPTCHA generation. Furthermore, in Eq. 6.3, the process $Train$ represents gradient descent learning. For a given set of parameters ϕ_t, let $E_{H,ad}(\phi_t)$ be the objective function that minimizes the error caused by four conditions associated with H and Ad:

$$E_{H,Ad}(\phi_t) = \begin{cases} 0 & \text{if } H = 1 \text{ and } Ad = 0 \\ 1 & \text{otherwise.} \end{cases} \tag{6.4}$$

Like with FaceDCAPTCHA, FR-CAPTCHA's gradient descent optimization process obtains optimal generation parameters using these two equations:

$$\bigtriangledown E_{H,Ad}(\phi_t) = \frac{\partial E_{H,Ad}(\phi_t)}{\partial \phi} \tag{6.5}$$

$$\phi_{t+1} = \phi_t - \eta \bigtriangledown E_{H,Ad}(\phi_t) \tag{6.6}$$

Here, $\bigtriangledown E_{H,Ad}(\phi_t)$ represents the gradient of the error function at the tth learning iteration. The variable η is the learning rate that controls the convergence of parameter learning. This learning procedure involves determining the human response (H) as well as the adversarial response for the automated face detection and face recognition algorithms (Ad); therefore, it converges to the case where humans can solve the CAPTCHA but the adversarial algorithms cannot.

Human responses were collected from participants attempting to complete the generated training CAPTCHAs. The Viola-Jones face detector [153] and a commercial off-the-shelf (COTS) face recognition system were used to provide adversarial responses using the following methodology:

1. Given a CAPTCHA image C, a set of images c_1 to c_n is created by rotating the CAPTCHA image by n incremental rotation levels ranging from 0° to 360° to model the random rotation of the embedded images.

2. For each CAPTCHA image c_1 to c_n, the Viola-Jones face detector [153] is used to perform face detection.

3. The detected face coordinates are matched with the actual embedded genuine face photograph locations. If an embedded face was correctly identified in any $c_1 \ldots c_n$ image, it is considered to be found for C.

4. If all embedded faces in C were found, the face detector's adversarial response is considered correct. Otherwise, it is incorrect.

5. Steps 2 through 4 are repeated using a COTS face recognition package in place of the Viola-Jones face detector. If the face recognition package was able to correctly detect two embedded faces, its face recognition module is invoked to attempt a match. The face recognition adversary is considered to be successful if it is able to find a correct pair of embedded faces belonging to the same individual.

Final CAPTCHA generation using optimized parameters

Once training is complete, the final optimized FR-CAPTCHA images are created using the parameters learned by the gradient descent optimization process. These optimized CAPTCHAs are generated by the same process described earlier in this section for the training CAPTCHA images. Examples of the final optimized FR-CAPTCHA images for each parameter set are shown in Fig. 6.5.

6.2 Experimental results and analysis

Over 3,000 participants assisted in evaluating FR-CAPTCHA. This section describes the images used in generating the CAPTCHA, the experimental protocol, and key results.

Figure 6.5: Examples of FR-CAPTCHA images [169]. These images are numbered using the set numbers in Tables 6.2 and 6.3.

6.2.1 Image databases

The images embedded in FR-CAPTCHA were taken from publicly available online images. Human face photographs were taken from the AR Face Database [173]. Fake cartoons, emoticons, and non-face distractor images were selected from Photobucket [162]. The fake images were selected because they generated false positives when they were analyzed by face detection algorithms. Of the source images, 40% were used in generating training images; the remaining images were used to generate the final optimized test CAPTCHAs.

6.2.2 Participants and testing protocol

Human accuracy in solving FR-CAPTCHA images was tested by over 3,000 people. To evaluate the 300 training images, a group of users was asked to repetitively complete as many FR-CAPTCHA images as they wished using a device of their own choosing. This facilitated gathering a large amount of data for training purposes in a short amount of time. Evaluation of the 500 final optimized images was completed using a much larger set of unsupervised volunteers who were asked to access a FR-CAPTCHA protected website. These users were asked to continue solving CAPTCHAs until they were successful.

Automated attack testing was performed using the Viola-Jones face detection algorithm and a COTS face recognition system. If the attackers were able to detect the embedded images needed to solve the CAPTCHA without any erroneous detections, their attack was considered successful.

6.2.3 Analysis

Training results

A total of 220 volunteers recorded 1,794 attempts while evaluating training FR-CAPTCHA images. Their overall success rate was 96%. The data shown in Table 6.2 provides valuable insight into the impact of applied distortions on human accuracy. In general, human performance is degraded as more intense distortions are applied to the generated images. Upon review of the results, the lowered human success rates for Sets 7 (79.2%) and 8 (90%) stand out. These are the only sets that use more than a low level of background blending. Like with other approaches, humans have difficulty solving this CAPTCHA's tests under conditions when the embedded images are heavily blended into the background.

In reviewing the remaining sets, rotation, global distortions, and emoticons appear to have a relatively smaller impact on human success rates. They are better distortions to emphasize when generating the final optimized FR-CAPTCHAs than background blending.

Table 6.2: Human performance on FR-CAPTCHA training images [169]

Set	**Distortions Used**	**Human Success**
1	No rotation, no blend, no global distortions, no emoticons	100%
2	Low rotation, no blend, no global distortions, no emoticons	100%
3	Low rotation, low blend, no global distortions, no emoticons	100%
4	Low rotation, low blend, low global distortions, no emoticons	100%
5	Low rotation, low blend, low global distortions, emoticons	100%
6	High rotation, low blend, low global distortions, no emoticons	100%
7	Low rotation, high blend, no global distortions, no emoticons	79.2%
8	Medium rotation, medium blend, no global distortions, emoticons	90.0%
9	Low rotation, low blend, low global distortions, no emoticons	100%
10	Low rotation, low blend, high global distortions, emoticons	92.0%

These distortions are overall much simpler and less intense than those used in previous CAPTCHAs such as fgCAPTCHA. The added difficulty that automated attackers face in completing the face recognition task enables use of lighter distortions without sacrificing security.

The automated attack adversaries were able to locate a small percentage of faces in the training sets with lower levels of distortions. They were unsuccessful in locating any genuine faces in more heavily distorted sets.

Human performance evaluation

The final optimized FR-CAPTCHA images were evaluated in 42,000 attempts recorded by over 3,000 participants. Overall, these participants achieved an extremely high 94% success rate in their attempts. While this rate is somewhat less than the 96% found during the training, this rate represents a more diverse user base and is representative of real world results. This 94% rate bests most other existing CAPTCHA solutions such as IMAGINATION, Kluever's Video CAPTCHA, and Adamas [174, 104, 65].

Table 6.3: Human and automated attack success rates for final optimized FR-CAPTCHAs

Set	Distortions Used	Human Success	Attack Success
1	No rotation, no blend, no global distortions, no emoticons	96.7%	0.0%
2	Low rotation, no blend, no global distortions, no emoticons	96.1%	0.0%
3	Low rotation, high blend, no global distortions, no emoticons	95.4%	0.0%
4	Medium rotation, low blend, no global distortions, no emoticons	96.1%	0.0%
5	Medium rotation, no blend, low global distortions, no emoticons	95.8%	0.0%
6	Medium rotation, low blend, low global distortions, no emoticons	94.7%	0.0%
7	High rotation, medium blend, medium global distortions, no emoticons	91.5%	0.0%
8	High rotation, low blend, medium global distortions, emoticons	93.5%	0.0%
9	High rotation, high blend, high global distortions, emoticons	90.0%	0.0%
10	High rotation, low blend, high global distortions, emoticons	91.5%	0.0%
Overall		94.0%	0.0%

As during the training phase, human success rates varied depending on the intensity of the distorted images. Table 6.3 provides a breakdown of success rates by distortion set. Set 1, which applied no distortions, had the highest human success rate at 96.7%. Set 9, which had high levels of all distortions, had the lowest success rate at 90% with other sets falling in between relative to the level of distortions applied.

Automated attack evaluation

Besides offering good human success rates, an effective CAPTCHA must also be resilient against automated attacks. Since the first step for an automated attacker in solving FR-CAPTCHA would be to detect and segment the embedded faces, face detection tests were conducted on the final optimized FR-CAPTCHA images using the Viola-Jones face detection algorithm and the COTS face recognition system used during training. These would-be

attackers were unable to correctly detect any embedded face images, which indicates they would be unable to solve FR-CAPTCHA. An existing attack strategy [93] used successfully against the IMAGINATION CAPTCHA was also tested, but it was similarly unable to locate the face pairs needed to solve FR-CAPTCHA.

Random guess-based brute force attacks are unlikely to succeed. Each FR-CAPTCHA contains four embedded face pair images (two images for each of two pairs), with the face region of each being approximately 100×100 pixels in size. Once the first image is selected, only one 100×100-pixel area out of the entire 600×400-pixel CAPTCHA that remains will result in a successful solution. Thus, we can represent the likelihood of any one attempt successfully solving an FR-CAPTCHA test as approximately 7 in 1000:

$$\left(\frac{4 \times 100 \times 100}{600 \times 400}\right)\left(\frac{100 \times 100}{600 \times 400 - 1}\right) = 0.6944\% \tag{6.7}$$

6.3 Summary

By combining gradient descent-optimized images with a face recognition-based task, FR-CAPTCHA succeeds in creating a security test that is easy for human users to solve while remaining resilient against automated attack now. Because of the complexity of face recognition, especially when images are distorted as in FR-CAPTCHA, it is expected that FR-CAPTCHA's tests will retain their effectiveness against would-be attackers well into the future.

Chapter 7

MB-CAPTCHA

7.1 Proposed approach

7.1.1 CAPTCHA design

Despite extensive research, the detection of face, fingerprint, and eye features embedded in larger images remains a challenging task for computers to perform. Humans, however, are adept at locating these items. *MB-CAPTCHA* [175] takes advantage of this relative difference in abilities by incorporating multimodal biometrics into its tests. The CAPTCHA presents users with complex images including embedded instructions along pictures of faces, fingerprints, and eyes as shown in Fig. 7.1. To solve MB-CAPTCHA, users must interpret the instructions (e.g., "Select all fingerprints and old faces") and select the embedded images that meet the provided criteria. If the user is able to correctly select the matching images without any mis-selections, they pass the CAPTCHA's test.

7.1.2 Generation process

The MB-CAPTCHA generation process can be represented as:

$$C = F(I_{face}, I_{eye}, I_{fingerprint}, \phi, d) \tag{7.1}$$

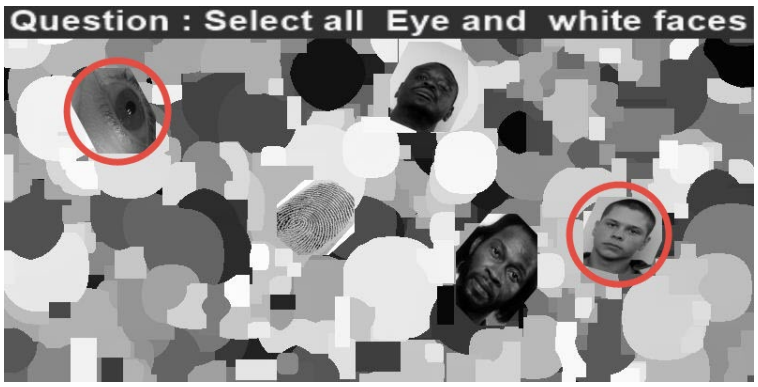


Figure 7.1: Sample MB-CAPTCHA image with correct answers circled in red.

where function F represents the series of image processing operations required to generate CAPTCHA C. C contains embedded images selected from a combination of sets I_{face}, I_{eye}, and $I_{fingerprint}$ depending upon a selection task chosen from ϕ. d represents a difficulty level from 1 to 5 used to determine distortions and image characteristics for the rendered CAPTCHA. CAPTCHAs with higher difficulty levels are intended to be more challenging to complete.

As shown in Fig. 7.2 and detailed below, several steps are involved in generating MB-CAPTCHA images.

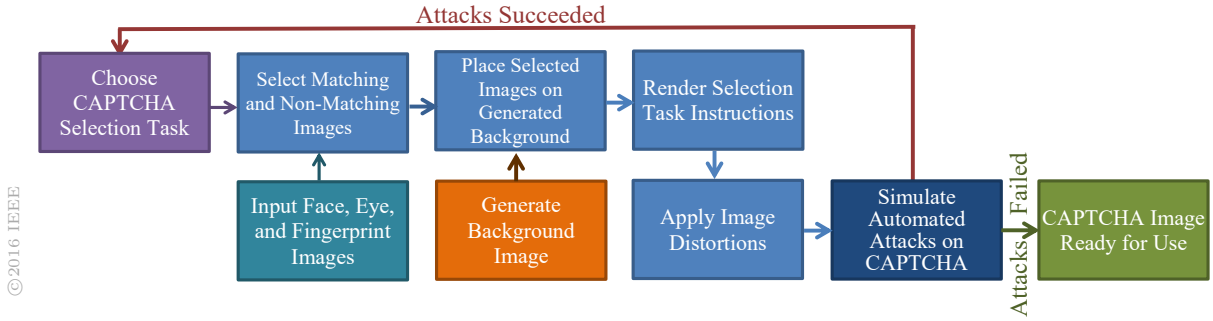


Figure 7.2: Steps involved in generating MB-CAPTCHA images [175]. A selection task is chosen to determine which items the user will be instructed to locate. Images matching the selection task and non-matching distractor images are then selected and placed on a generated background. Instructions are included in the image and then distortions are applied to the CAPTCHA. Simulated attacks are conducted. If a simulated attacker can solve the CAPTCHA, the generated image is discarded and a new one generated in its place. Otherwise, the generated MB-CAPTCHA image is ready for use.

Background generation

The generation of new MB-CAPTCHA tests begins with the creation of an 800×400-pixel image containing a randomly shaded grayscale background. Between 900 and 1,500 geometric shapes, either circles, rectangles, or crosses, are semi-transparently overlaid in random locations. Each individual shape is of a randomly selected size and grayscale shade. This complex pattern is intended to create false targets to confuse object detection algorithms that may be used to attack the CAPTCHA. During the testing conducted as part of this research, automated algorithms detected many false faces, fingerprints, and eyes in the background that did not actually exist. Since mis-selections are treated as incorrect attempts, these false images reduce the likelihood that an attack on the CAPTCHA is successful.

After all of the geometric shapes are placed, dilation is repeatedly performed on the entire background. This operation sets each pixel to the maximum (lightest) value of its adjoining pixels. It has the effect of creating a ragged, irregular border instead of crisp lines and reduces the success rate of edge detection and segmentation-based attacks. Fig. 7.3 shows an example of a rendered background after dilation is performed.

The dilation operation can be represented as [176]:

$$I \oplus S = \bigcup_{p \in I} S_p \tag{7.2}$$

where I is the image being dilated, S is a 3×3 structuring element, and S_p is the value of the structuring element centered at the pixel location p.

Selection task assignment

To solve a MB-CAPTCHA, users are required to select the embedded images that meet the requirements of a specified selection task. This step determines the selection task that will be used for a particular CAPTCHA image.

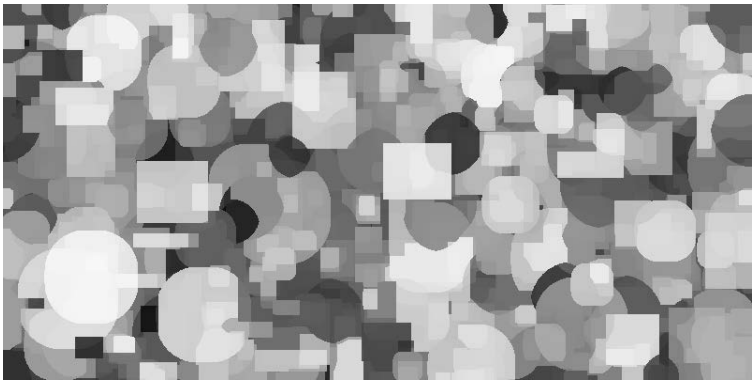


Figure 7.3: Example of a rendered background after dilation [175].

The process begins by randomly choosing to use either one or two selection tasks for the CAPTCHA. Next, the generation algorithm randomly picks specific tasks from these three options:

1. Eye selection, where users must locate all of the images of eyes embedded in the CAPTCHA.

2. Fingerprint selection, where users must locate all of the images of fingerprints embedded in the CAPTCHA.

3. Face selection, where users are asked to select all embedded face photographs matching a specified gender (male or female) or race (black or white) attribute. The attributes are selected at random.

If two selection tasks are to be used for the CAPTCHA, the generation algorithm ensures two different tasks are selected.

Image selection

After the selection tasks are chosen, the generation algorithm constructs two sets I_{match} and $I_{notmatch}$ based on images that match and do not match, respectively, the CAPTCHA's selection task requirements. Fig. 7.4 shows examples of the source eye, fingerprint, and face images that are organized into sets I_{match} and $I_{notmatch}$.

Figure 7.4: Examples of source images that will be embedded in generated MB-CAPTCHAs [175].

A total of four to eight matching and non-matching images are selected from sets I_{match} and $I_{notmatch}$ for use in the CAPTCHA so that:

$$n_{total} = \left\{ n_{match} + n_{notmatch} \;\middle|\; n_{match} \geq 2, n_{notmatch} \geq 2, 4 \geq n_{total} \geq 8 \right\} \tag{7.3}$$

n_{match}, $n_{notmatch}$, and n_{total} represent the number of matching, non-matching, and total embedded images, respectively. At least two matching images are required to ensure that a single guess cannot solve the CAPTCHA. Multiple non-matching images serve as false targets to reduce the likelihood that an automated algorithm will correctly solve the CAPTCHA.

When face images are used, the images contain a background surrounding the face when the difficulty level is $d \leq 3$. There is no background surrounding the face when $d > 3$.

Image placement

Each of the images to be embedded in the CAPTCHA is scaled to between 80×80 and 120×120 pixels in size to ensure there is room to embed the required number of images within the CAPTCHA. The images are also rotated at randomly selected amounts between

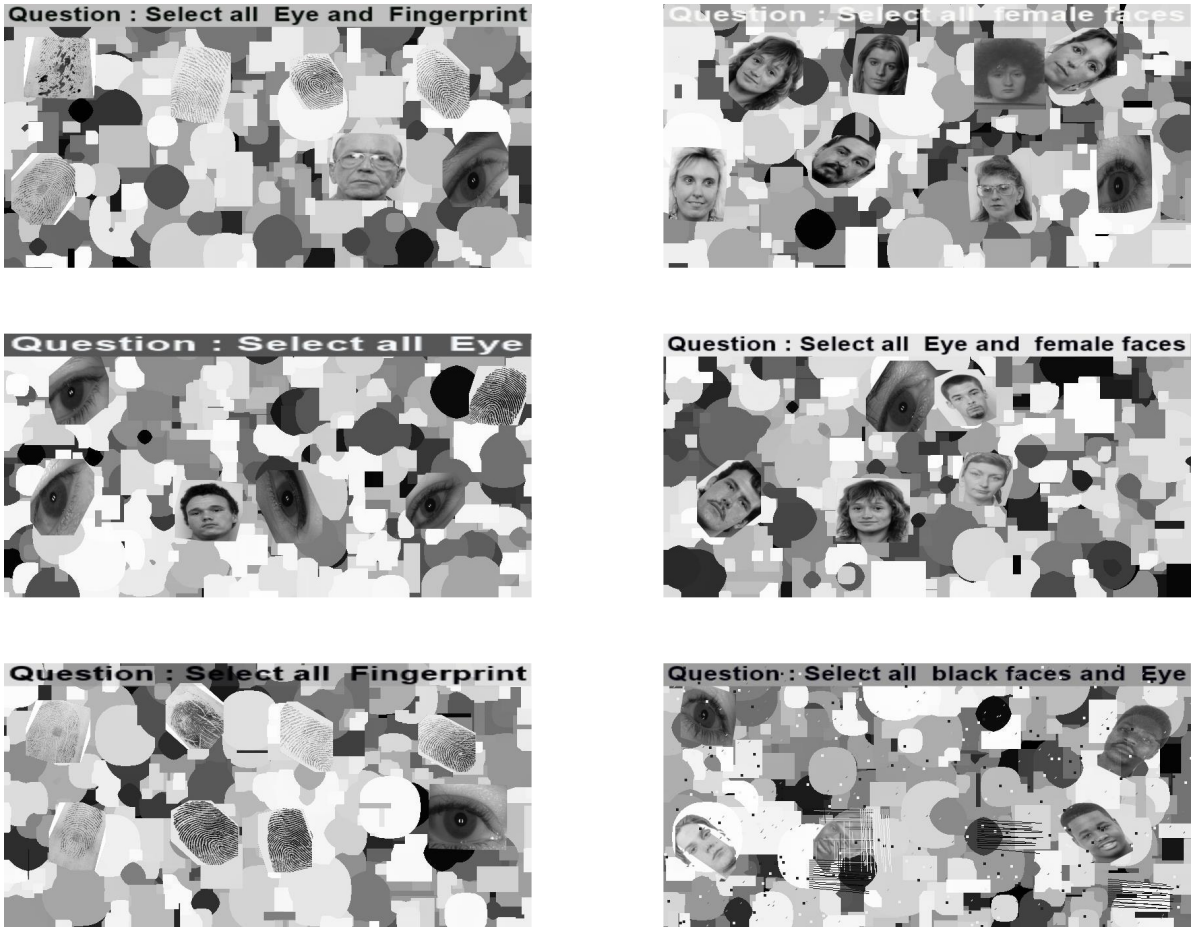


Figure 7.5: Samples of MB-CAPTCHA images [175]. Users must select the images specified in the instructions to correctly solve the CAPTCHA.

$[-60, 60)$ degrees. The rotation decreases the likelihood that detection algorithms will correctly identify the images once embedded in the background.

The image placement algorithm randomly selects locations for the embedded images. In choosing locations for placement, the algorithm ensures that the embedded images do not overlap with each other or the edge of the CAPTCHA. It also maintains a 35-pixel margin at the top of the CAPTCHA for the selection task instructions to be placed.

Each image is embedded using alpha compositing (partial transparency) so it blends into the background. This blending interferes with object detection since the embedded images

may have different contrasts, colorings, and edges than what the algorithms are trained to detect.

Instructions rendering

Once all of the images have been placed, instructions for solving the CAPTCHA are rendered. The instructions are displayed using the Arial font on a randomly shaded grayscale background with a randomly selected alpha channel (transparency) value. The color of the text varies upon the shade of the background: white text is used for darker backgrounds and black text for lighter backgrounds. For cases where the difficulty level is $d = \{1, 3, 5\}$, the text is additionally blurred to decrease the likelihood of optical character recognition (OCR) being able to identify the selection task needed to complete the CAPTCHA.

The rendered text image is stretched to fit an area of 800×35 pixels, as shown in the examples in Fig. 7.6. It is then embedded at the top of the CAPTCHA using alpha compositing. In cases where the alpha value is low, the grayscale shading appears transparent and the underlying complex background pattern shows through the text. This greatly reduces the success rate of OCR attacks in correctly identifying the selection task required for the CAPTCHA.

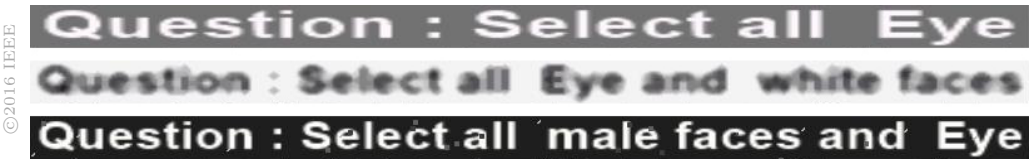


Figure 7.6: Examples of instructions for solving MB-CAPTCHA prior to being rendered in the CAPTCHA image [175].

Image distortion

Depending upon the difficulty level d used in generating the CAPTCHA, additional visual effects may be applied to the rendered CAPTCHA to distort the image. When $d = \{3, 5\}$,

lines in various shades of gray may be drawn in random locations throughout the CAPTCHA in a striped or crosshatch pattern. Many small randomly shaded grayscale squares are also placed on the image when $d = \{3, 5\}$. These additional distortions are intended to interfere with the ability of detection algorithms to locate and segment the images embedded in the CAPTCHA or to perform OCR on the instructions. While the distortions have some negative impact on human users, it is minimal compared to the effect on automated attackers.

Attack simulation

After each CAPTCHA image is rendered, it undergoes a series of tests designed to ensure that it is not susceptible to attack by automated algorithms. The first test involves using best-of-breed commercial OCR in an attempt to detect the selection task specified by the CAPTCHA's instructions. If the selection task can be correctly identified, further testing is required to ensure the CAPTCHA's embedded images cannot be automatically detected:

1. If the selection task requires that faces be identified, the Face++ web service [177] is used to locate embedded faces. This tool determines the likely race and gender for each face it locates. The detected values are compared against the actual embedded faces to see if the required faces were correctly found.

2. If the selection task requires that eyes be identified, a version of the Viola-Jones algorithm [153] trained for eye detection is used to locate them. The detected locations are compared against the actual eye locations.

3. If the selection task requires that fingerprints be identified, the SourceAFIS fingerprint recognition software [178] is used to construct fingerprint templates based on the image. The location of the templates' features are compared against the actual locations of the embedded fingerprints to see if they match.

Face++, Viola-Jones, and SourceAFIS are used as they can be performed in near real time. Fast performance at this stage is critical since these tests are run in-the-loop during the CAPTCHA generation process, and as such, occur frequently.

In the event these tests are able to correctly determine the selection task and identify the images required to solve the CAPTCHA, the CAPTCHA image is discarded and the generation process begins anew. This ensures that publicly used CAPTCHA test images are resistant to automated attacks and viable as a security tool. Fig. 7.5 shows examples of CAPTCHAs that have passed the simulated attacks and are ready for use.

7.2 Experimental results and analysis

MB-CAPTCHA has been evaluated by over 1,900 human users. This section provides the details of the source image databases, participants, and protocol used in evaluating the CAPTCHA along with results and analysis.

7.2.1 Image databases

MB-CAPTCHA uses three source databases to provide the face, fingerprint, and eye images used in generating the CAPTCHAs: the University of North Carolina Wilmington Craniofacial Morphology Database is used for face images [179], eye images are from the IIITD Multi-spectral Periocular Database [180], and fingerprint images are from the FVC2004 database [181].

7.2.2 Participants and testing protocol

MB-CAPTCHA was evaluated by 1,905 participants using a set of 473 rendered CAPTCHA images. Participants attempted to access a website protected by MB-CAPTCHA in an uncontrolled environment on a device of their choosing. Users were asked to continue attempting to solve the CAPTCHA until their attempts were successful.

7.2.3 Analysis

Human performance evaluation

Participants recorded a total of 30,664 attempts at solving MB-CAPTCHA. The overall human success rate across all selection tasks was 83.6%, but Table 7.1 shows there was significant variation in success rates depending upon the exact selection task used. Humans were best able to solve CAPTCHAs where they were just asked to locate embedded items, such as eyes and fingerprints, rather than tasks where they had to both locate and categorize items, such as when selecting faces of specified genders or races. Users could generally identify the location of faces but sometimes failed to properly categorize them. Part of this difficulty may arise from ambiguity in the embedded images. For example, more than half of the users failed to identify the circled face in Fig. 7.7 as being a man.

Figure 7.7: Many people did not accurately identify the gender of the circled face [175].

The CAPTCHAs' difficulty level and associated distortions seem to have minimal impact on humans' ability to successfully solve them. One situation where distortions do appear to hinder humans is when crosshatched lines or rectangles are placed on top of embedded images, as sometimes happens in CAPTCHAs with a difficulty level of $d = \{3, 5\}$. These distortions can make it difficult to detect the embedded object, or in the case of faces, to correctly identify their attributes. Fig. 7.8 shows an example where a rectangle placed over

Table 7.1: MB-CAPTCHA success rates by selection task [175]

Selection Task	Human Success	Attack Success
Eyes	90.5%	0.0%
Fingerprints	89.4%	0.0%
Faces of specified race, eyes	87.9%	0.0%
Fingerprints, eyes	83.7%	0.0%
Faces of specified race	80.0%	0.0%
Faces of specified gender and race	72.7%	0.0%
Faces of specified gender	72.1%	0.0%
Faces of specified gender, fingerprints	67.1%	0.0%
Faces of specified gender, eyes	63.8%	0.0%
Faces of specified race, fingerprints	62.8%	0.0%
Overall	**83.6%**	**0.0%**

an embedded image caused many users to fail to select the circled face. Otherwise, the impact of the difficulty levels and distortions was slight.

Figure 7.8: Many users failed to select the circled face [175]. The overlaid rectangle makes it difficult to determine the person's gender.

Automated attack evaluation

Since MB-CAPTCHA's generation process is designed to remove images susceptible to object detection-based automated attacks, one of the most likely remaining avenues of attack

involves brute force random guessing. Each CAPTCHA contains two to six images that must be selected, each on average about 100×100 pixels in size, from an area 800×400 pixels in size. The chance of a single random guess at correctly answering the CAPTCHA is microscopic:

$$\left(\frac{1}{5}\right) \sum_{i=2}^{6} \left(\prod_{j=1}^{i} \frac{(100 \times 100)j}{(800 \times 400) - (i - j)} \right) = 0.0433\% \tag{7.4}$$

As previously shown CAPTCHA test images are not repeated until the set of all unseen images have been exhausted, it is unlikely that a would-be attacker would see the same CAPTCHA test again for some time after a failed attempt. When combined with the less than 1-in-2,000 chance of correctly solving an individual attempt, it becomes extremely time-consuming to use a brute force approach to defeat MB-CAPTCHA.

7.3 Summary

MB-CAPTCHA represents a different approach to image-based CAPTCHAs than the previous face-based designs. By incorporating three biometric modalities (eye, fingerprint, and attribute-specific face recognition), MB-CAPTCHA provides a solution that is significantly more challenging for automated attackers to solve while providing relatively high human success rates that are comparable to those offered by existing solutions like the 2017 version of DeepCAPTCHA [139] and the MSN CAPTCHA [22].

Chapter 8

aiCAPTCHA

8.1 Proposed approach

8.1.1 CAPTCHA design

Compared to the previously covered approaches, *aiCAPTCHA* [182] uses a wider set of embedded objects to provide greater resiliency against improved face detection and recognition attack strategies [183, 184]. As illustrated in Fig. 8.1, aiCAPTCHA presents users with a composite image containing a stack of photographs of items representing 100 classes and 172 attributes (e.g., green tractors). Users are instructed to select specific items present in the CAPTCHA image, such as black chairs or red tomatoes. If the user is able to select all of the required objects, allowing one mistake where they fail to select a genuine object or select an incorrect object, their attempt is counted as correct.

While object recognition tasks similar to those used in aiCAPTCHA have been extensively studied in computer vision [185, 186, 187], existing solutions remain inferior to the human visual system and are generally optimized for a handful of object types; they perform poorly when used with extensive object classes like those used by aiCAPTCHA. To further ensure that automated attackers are unable to solve aiCAPTCHA tests, the generation process incorporates a negative selection-based artificial immune system that identifies and

Figure 8.1: Example of an aiCAPTCHA test based on identifying white flowers. The correct answers are circled in red.

removes CAPTCHAs found to be susceptible to automated attack. This process ensures a good balance between ease-of-use for legitimate users and resiliency against automated attackers.

8.1.2 Generation process

The aiCAPTCHA generation process can be represented as:

$$C = F(width, height, O, I, H, d, A) \tag{8.1}$$

where function F represents the series of operations required to generate a new aiCAPTCHA of dimensions *width*-by-*height* pixels. The required object type used for the CAPTCHA test is randomly chosen from O, the set of all tagged object classes and attributes. True target and false target photographs are taken from object-tagged set I. H contains precomputed Histogram of Oriented Gradient (HOG) descriptors for each photograph in I, an alternate representation of image data that can be used to identify similar images [188]. d repre-

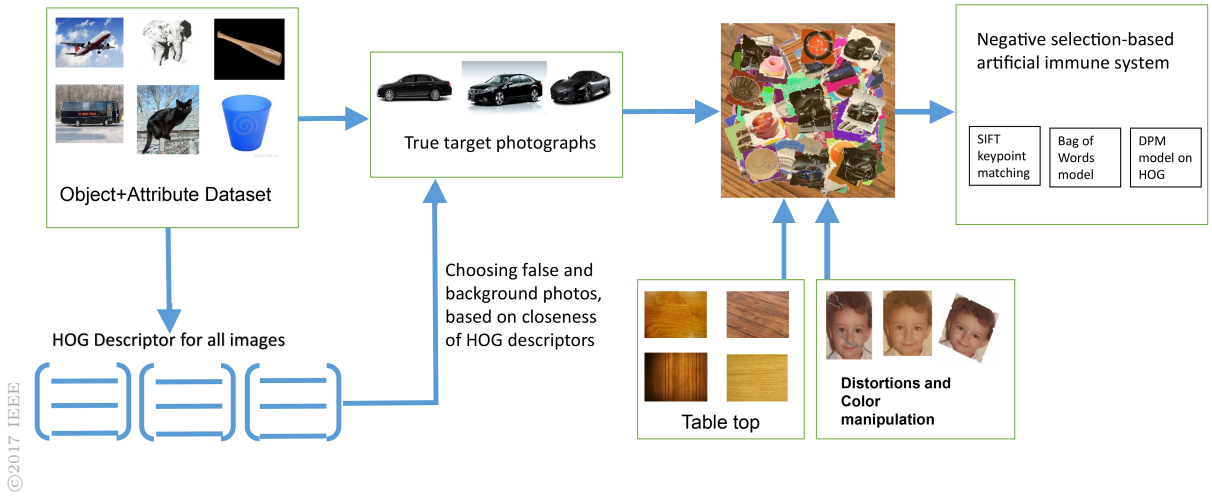


Figure 8.2: An overview of the aiCAPTCHA generation process [182]. The process begins with an attribute-tagged set of images and accompanying set of HOG descriptors. The true target photographs users will be required to select are chosen, and then false and background distractor images are selected based on the relationship of their HOG descriptors to the selected true target photographs. The images are placed on a generated table top image and distortions are applied. A negative selection-based artificial immune system conducts simulated attacks on the generated CAPTCHAs to remove images that are vulnerable to attack from the set of public aiCAPTCHAs.

sents a difficulty level in the range $[1, 4]$ that determines the distortions added to generated CAPTCHAs and A represents the attack algorithms to be used in conducting negative selection and deletion of vulnerable CAPTCHAs. The resulting attack-resistant generated aiCAPTCHA is C.

As shown in Fig. 8.2, a number of steps are involved in the generation of aiCAPTCHA images. They are described below.

Background generation

Generation of a new aiCAPTCHA image begins with the creation of a background of size $width \times height$ pixels, where large sizes (at least 750×750) are preferred to improve image quality on high resolution displays and to minimize the likelihood of a successful attack. The

background is designed to resemble a table top upon which a stack of photographs will be placed.

Target object type selection

Once the background is generated, one object class-attribute combination (object type), designated $o_{selected}$, is randomly chosen from O to use for the CAPTCHA test. This target object type will determine which photographs are embedded in the aiCAPTCHA image and will be provided to users in the instructions to solve the CAPTCHA.

True target photograph selection

Using the chosen target object type $o_{selected}$, between three and five images are randomly chosen from the subset of corresponding photographs $I_{d_{selected}}$. These images, represented by P_{true}, will be embedded in the resulting aiCAPTCHA image and serve as true targets to be selected when solving the CAPTCHA.

False target photograph selection

Next, false target photographs are chosen to function as distractors for a would-be attacker. False targets are chosen by first calculating the Euclidean distance between the HOG descriptor for each photograph in P_{true} relative to each photograph in $I_{notselected} = \{I - I_{o_{selected}}\}$. The photographs with a smaller Euclidean distance in their HOG descriptors are more visually similar and are ideal for use in the CAPTCHA. A would-be attacker may be more likely to confuse one visually similar photograph for another and thus fail at solving aiCAPTCHA.

For each true target photograph in P_{true}, the three to four photographs in $I_{notselected}$ with the smallest Euclidean distance between their HOG descriptors are added to P_{false} to serve as false targets in the aiCAPTCHA image.

An additional 10 to 20 randomly selected photographs from $I_{notselected}$ are added to $P_{background}$ to provide a confusing background upon which P_{true} and P_{false} will be layered.

Photograph preparation and placement

After selecting $P_{background}$, P_{true}, and P_{false}, each photograph in these sets is rotated and scaled as follows [158]:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x cos\theta & -s_y sin\theta & 0 \\ s_x sin\theta & s_y cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{8.2}$$

Here, s_x and s_y are the scaling factors in the x and y directions respectively and θ is the clockwise rotation angle. The scaling parameters are determined based on the individual size of each photograph, and θ is varied randomly within a range carefully chosen to avoid extremely low or extremely high rotations. (x', y') denote the new coordinates for each pixel (x, y) of each photograph. The scaling allows for consistent sizing of each photograph (approximately 100×100 pixels) so it can be easily viewed. The rotation helps achieve the desired output in the form of a stack of photographs.

After scaling and rotation, the photographs may be subjected to the tear distortion when difficulty level $d = \{3, 4\}$. The tear distortion is applied to each photograph individually by randomly selecting one or more pairs of points on the image. For each of these pairs, one point is designated as the starting point (x_0, y_0) and the other is designated as the ending point (x_f, y_f). A number of lines, each denoted by T (initialized with the starting point (x_0, y_0)), are constructed between each starting and ending point as follows:

$$T = T \bigcup (\frac{x_f - x_t}{x_f} r_x(\cdot), \frac{y_f - y_t}{y_f} r_y(\cdot)) \tag{8.3}$$

Here, (x_t, y_t) denotes the latest point added to the line T, (x_f, y_f) denotes the ending point for this line. $r_x(\cdot)$ and $r_y(\cdot)$ both denote random functions that are either 0 or 1 but with the constraint that for each step $r_x(\cdot) + r_y(\cdot) \geq 1$. Therefore, at each step, Eq. 8.3 takes a one pixel step towards the ending point but it may choose from three possible directions.

Each line T starting from the same pair of starting and ending points is added to the set of tear lines $\mathbf{T}$ and the pixels of the image for each point in this set ($(x, y) \in \mathbf{T}$) are modified as follows:

$$I'(x, y) = \Big\{ t + \delta | \delta \in [-v, +v] \Big\} \tag{8.4}$$

where $I'(x, y)$ denotes the new value for the pixel at location (x, y) for the particular image I. t denotes the base tear color (either gray or white). The variable δ represents a small change in the range of $[+v, -v]$ where v denotes the maximum permissible deviation from the base tear color.

Next, a random walk function is used to create a ragged edge effect around each photograph. Finally, the colors in the photographs are manipulated to make object recognition more challenging.

After all the visual effects are added, each photograph in $P_{background}$ is placed layer by layer at random locations on the background. Next, the photographs in P_{true} and P_{false} are placed in the top-most layer so that the true target P_{true} photographs are always completely visible to the user.

Global distortion

Once the entire aiCAPTCHA image is generated, a dust effect is applied globally for difficulty levels $d = \{2, 4\}$. The goal of the dust effect is to make aiCAPTCHA harder to solve for automated attackers while adding a type of noise that is commonly seen by human eyes, i.e., the settling of dust. In order to emulate this effect, we divide the aiCAPTCHA image into regions and apply a blending effect to the pixels belonging to a region as follows:

$$I'(x, y) = w_i I(x, y) + w_d D \tag{8.5}$$

Here, $I'(x, y)$ and $I(x, y)$ denote the modified and original values of the pixel at location (x, y) of the aiCAPTCHA image, respectively. D is the dust color that is set at (242, 168, 0) RGB. w_i and w_d are the weights for the original pixel and the dust color D used in the weighted sum-based blending approach so that $w_i + w_d = 1$ and w_d varies between 0.1 to 0.3 depending on the image region.

Negative selection and deletion of vulnerable CAPTCHAs

To ensure the viability of aiCAPTCHA as a security tool, it is important that all of the images presented to users are resilient to automated attacks. Much as biological immune systems use a negative selection process to identify and eliminate immune cells that do not properly guard against foreign attackers, aiCAPTCHA employs an artificial immune system with its own negative selection process to identify and remove aiCAPTCHA images that may not successfully protect the guarded resource from automated attack [189]. Rather than use generated detectors as in a traditional negative selection algorithm [190], aiCAPTCHA's process uses input from three distinct object recognition algorithms as simulated attackers to determine which aiCAPTCHA images should be removed:

1. SIFT (Scale-Invariant Feature Transform) keypoint-based matching [191, 192], which generates feature-based descriptions of images. The features generated from the aiCAPTCHA images are compared to already tagged image templates representing known objects using a 0.6 cosine similarity threshold. If the threshold is met, the object identified in the aiCAPTCHA is labeled.

2. Bag of Visual Words image classification [193], which constructs a sparse vector of histograms representing image features and then uses a Naive Bayes classifier to attempt to match those to the feature histograms of previously trained images.

3. Discriminatively-trained deformable part-based model (DPM) categorization [194, 195], which builds multiscale deformable models representing portions of images. Support

vector machines are used to match the models generated from aiCAPTCHA images to previously trained known models of objects.

Simulated attacks are conducted against generated aiCAPTCHA images using each of the three algorithms. If any algorithm successfully identifies the objects in at least half of the true target photographs, the CAPTCHA is considered to be defective and is deleted from the aiCAPTCHA database by the negative selection algorithm. This ensures that the resulting aiCAPTCHA images are resilient to adversarial external attacks. As shown by Table 8.1, approximately 14% of generated aiCAPTCHA images were deleted by the negative selection algorithm while conducting this research. CAPTCHA with no distortions or just dust-type distortions were most likely to be found by the automated attackers; addition of the tear distortion reduced the likelihood of a successful automated attack by one-third. Fig. 8.3 shows examples of aiCAPTCHA images that have passed the negative selection attack process and are ready for use.

Table 8.1: Number of aiCAPTCHAs solved by attackers in negative selection artificial immune system while generating images [182]

Generated CAPTCHAs	**860**
CAPTCHAs solved by SIFT	33
CAPTCHAs solved by Bag of Visual Words	0
CAPTCHAs solved by DPM	91
Defective Attackable CAPTCHAs Deleted from Database	**124**
Remaining Resilient CAPTCHAs for Public Use	**736**

8.2 Experimental results and analysis

Generated aiCAPTCHA images were tested by over 3,000 human participants. This section provides details of the source images, research participants, and protocol used in evaluating aiCAPTCHA along with results and analysis.

Figure 8.3: Four examples of aiCAPTCHA images that passed the negative selection-based filtering process [182]. The true target photographs needed to solve the CAPTCHA are outlined.

8.2.1 Image databases

To generate aiCAPTCHA images, a database with attribute-labeled images of various object categories is required. Since existing object databases are either: (a) not labeled with attributes, or (b) restricted to specific groups of objects such as animals, a new database was collected to support this research. Collection began with the creation of a list of object classes and associated attributes for each class. For example, cats may have the attributes "white," "gray," or "black" based on color, whereas books may be "open" or "closed" depending on their position. Overall, 100 classes with 172 attribute-based subclasses were identified.

The identified classes and attributes were used to generate search queries to retrieve images for each combination of object class and attribute. The retrieved images were manually filtered to remove images that did not accurately represent the intended object class and attribute. The resulting database contains a total of 7,765 tagged images.

8.2.2 Participants and testing protocol

aiCAPTCHA was evaluated by 2,283 participants using a set of 736 rendered CAPTCHAs from the aiCAPTCHA and aiCAPTCHA+UX databases. Participants attempted to access portions of a website protected by aiCAPTCHA. Participants were unsupervised and allowed to use their choice of browser and computing device (desktop computer, laptop, tablet, or

smartphone). One aiCAPTCHA image was presented at a time. Users were asked to continue attempting to solve the CAPTCHAs until their attempts were successful.

Initially, all CAPTCHAs which pass the negative selection filtering process are placed in the aiCAPTCHA database. Over time, the CAPTCHAs that demonstrate a good user experience by achieving a 90% human success rate over the first 10 attempts are migrated to the aiCAPTCHA+UX database; the CAPTCHAs not meeting this threshold are discarded. This novel adaptive filtering mechanism ensures that CAPTCHAs in the aiCAPTCHA+UX database are both resilient to external attacks and provide an excellent user experience as quantitatively determined by user performance.

8.2.3 Analysis

Human performance evaluation

In total, 30,155 attempts to solve aiCAPTCHA were recorded with 7,360 attempts from CAPTCHAs in the aiCAPTCHA database and 22,795 attempts from CAPTCHAs in the aiCAPTCHA+UX database. As shown in Table 8.2, humans achieved a 94.6% success rate (correct 19 of 20 times) when attempting CAPTCHAs in the aiCAPTCHA+UX database.

Table 8.2: Success rates for aiCAPTCHA databases [182]

Database	**Number of CAPTCHAs**	**User Attempts**	**Human Success**	**Attack Success**
aiCAPTCHA	736	7,360	80.7%	0.0%
aiCAPTCHA+UX	410	22,795	94.6%	0.0%

The tested aiCAPTCHAs came from four distinct difficulty levels, each with its own set of distortions applied during the generation process. Use of these levels, shown in Table 8.3, can be determined by the application where the aiCAPTCHAs are deployed and the security needed. Overall, the impact of the difficulty levels and distortions on human success rates

Table 8.3: Evaluating success rates on aiCAPTCHA database images by difficulty level [182]

Difficulty Level	Distortions Used	Attempts	Human Success
Level 1	No dust or tear distortions	1,770	82.8%
Level 2	Dust distortions only	1,770	79.3%
Level 3	Tear distortions only	1,920	81.3%
Level 4	Both tear and dust distortions	1,900	79.0%

was small. Humans performed about 4% better on aiCAPTCHA database images with no distortions than those with both the tear and dust distortions.

Automated attack evaluation

The aiCAPTCHA generation process is designed to be resilient against attacks by conventional image classifiers through its artificial immune system. Unconventional and brute force attacks remain possible although testing with best-of-breed approaches finds them unlikely to succeed. Fig. 8.4 illustrates the results of attempting one such unconventional attack, image recognition with Very Deep Convolutional Networks [196], on two aiCAPTCHA images. As the figure shows, the algorithm was unsuccessful in correctly identifying the objects of interest in the CAPTCHAs.

Brute force attempts, where an attacker selects random locations in the CAPTCHA, are similarly likely to fail. Each aiCAPTCHA contains three to five true target photographs. Since one target is allowed to be missed in a successful attempt for the sole purpose of an improved user experience, an attacker would need to correctly identify between two and four targets to solve the CAPTCHA. Each target is approximately 100×100 pixels in size, with the overall aiCAPTCHA image size being at least 750×750 pixels. Thus, the average chance of a single brute force attempt at correctly solving aiCAPTCHA is extremely small:

$$\left(\frac{1}{3}\right) \sum_{i=2}^{4} \left(\prod_{j=1}^{i} \frac{(100 \times 100)j}{(750 \times 750) - (i - j)} \right) = 0.0223\% \tag{8.6}$$

Figure 8.4: Deep learning strategies were unsuccessful at correctly identifying the objects embedded in aiCAPTCHA images needed to solve the CAPTCHAs [182]. The labels atop each image above list the objects identified by Very Deep Convolutional Networks as being present in the image. None were accurate.

Since previously shown aiCAPTCHA test images are not repeated until all other unseen images have been exhausted, a would-be attacker would likely have to wait hundreds of times before they will see the same aiCAPTCHA image again if their attempt is unsuccessful. When combined with the 2-in-10,000 chance of correctly solving the CAPTCHA on a given attempt, a would-be attacker would likely spend a significant amount of time using a brute force approach to successfully attack aiCAPTCHA.

8.3 Summary

aiCAPTCHA represents a novel combination of an attribute-based object recognition CAPTCHA with a negative selection-based artificial immune system and two-phase filtering model that provides a security mechanism effective at preventing automated attacks without compromising the user experience. aiCAPTCHA has a high 94.6% human success rate for attempts

on CAPTCHAs in the aiCAPTCHA+UX database, well above the 70%-80% rate of existing CAPTCHAs such as reCAPTCHA and IMAGINATION [22, 97], and is designed to facilitate use on both traditional computers and touchscreen devices. When combined with its near zero likelihood of successful attacks, it offers significant advantages over CAPTCHAs commonly employed today.

Chapter 9

CAPTCHA Evaluation Platform

9.1 Purpose

Evaluation of the CAPTCHAs proposed in this research has required extensive human testing. The information collected from attempts at solving the 9,715 CAPTCHA images generated by this research (including click locations, answer timing, and browser environment details) represents several hundred megabytes of data. Management of the testing process and analysis of the resulting data has required the development of an extensive evaluation platform.

The CAPTCHA Evaluation Platform was developed to fulfill several key requirements:

1. **Allow for customized selection of CAPTCHA tests for each user attempt.** Every time a CAPTCHA is requested, a new CAPTCHA of the appropriate type must be selected for the particular user. To allow for rapid testing of a large number of CAPTCHAs, and also to increase resiliency against automated attacks, it is important that the same CAPTCHAs not be repeatedly displayed for a given user.

2. **Facilitate display of selected CAPTCHAs.** Once a CAPTCHA has been selected, the user's computer must be able to load the images and resources required to complete

it. The platform must guard against attempts to undermine the CAPTCHA's security benefits by preventing diversion of CAPTCHA tests to third parties [197].

3. **Collect data on attempts and user computing environments.** In order to evaluate each attempt at solving a CAPTCHA, the locations users select in the CAPTCHA must be recorded. Additional information on the user's computing environment and responses are also logged to facilitate in-depth analysis and understanding of CAPTCHA performance.

4. **Validate attempts at solving CAPTCHAs.** Each attempt must be validated to determine if it was a correct solution. The platform must further ensure that the attempt was recorded by the intended user and not by a third party.

5. **Analysis of collected result sets.** The platform must provide tools to view and analyze the collected data on a variety of metrics to allow for understanding of how the CAPTCHAs performed and how they can be improved.

9.2 Proposed approach

9.2.1 Platform design

The proposed platform has two major components: (a) a publicly accessible testing module to facilitate use of CAPTCHAs and (b) an internal analysis module to analyze data collected from CAPTCHA use.

The testing module is designed as a RESTful web service, a type of application programming interface (API) where clients use stateless HTTP requests to interact with a server [198]. By sending properly formatted HTTP GET and POST requests to defined URLs, clients, such as browsers and servers hosting CAPTCHA-protected webpages, can request new CAPTCHAs, load CAPTCHA images, and submit recorded data to determine if user attempts were valid.

The CAPTCHA web service is written in the C# programming language using the Microsoft ASP.NET Core 1.0 web application framework [199]. It is run on a Microsoft Windows Server 2012 R2 server with Microsoft Internet Information Services 8.5; this same server also stores the CAPTCHA images and supporting files. A large multi-table SQL database running on Microsoft SQL Server 2012 manages information on the available CAPTCHAs and stores records on each attempt.

Two code libraries were developed to support interaction with the CAPTCHA web service by browsers and web servers, respectively. The browser library is written in JavaScript and runs natively within all modern browsers. It is responsible for requesting and displaying CAPTCHA images. It also records each click or tap made in solving the CAPTCHA along with details on the browser environment. This information is stored in a hidden form field for validation. The browser library makes use of three external JavaScript libraries (JQuery [200], jCanvas [201], and Bowser [202]) to manipulate webpages and interact with the web service, to display an indicator on the locations where users have already clicked, and to record browser environment details, respectively.

The second code library is for use on servers hosting CAPTCHA-protected webpages. These servers must be able to determine if a would-be user correctly solved their CAPTCHA before granting access to protected resources. The C#-based server library submits the CAPTCHA attempt data collected from each user's browser to the web service for validation, and then processes the result returned by the web service.

The analysis module is designed as a series of extensions to the Microsoft SQL Server 2012 database used by the testing module. These extensions calculate a variety of statistics that can be used to measure the performance of tested CAPTCHAs on different characteristics such as their embedded images, distortion types, and difficulty levels. Database views are also provided to present raw CAPTCHA and attempt data in a de-normalized form to allow further analysis with external tools such as Microsoft Access, Microsoft Excel, and R.

9.2.2 Method of operation

Displaying CAPTCHAs

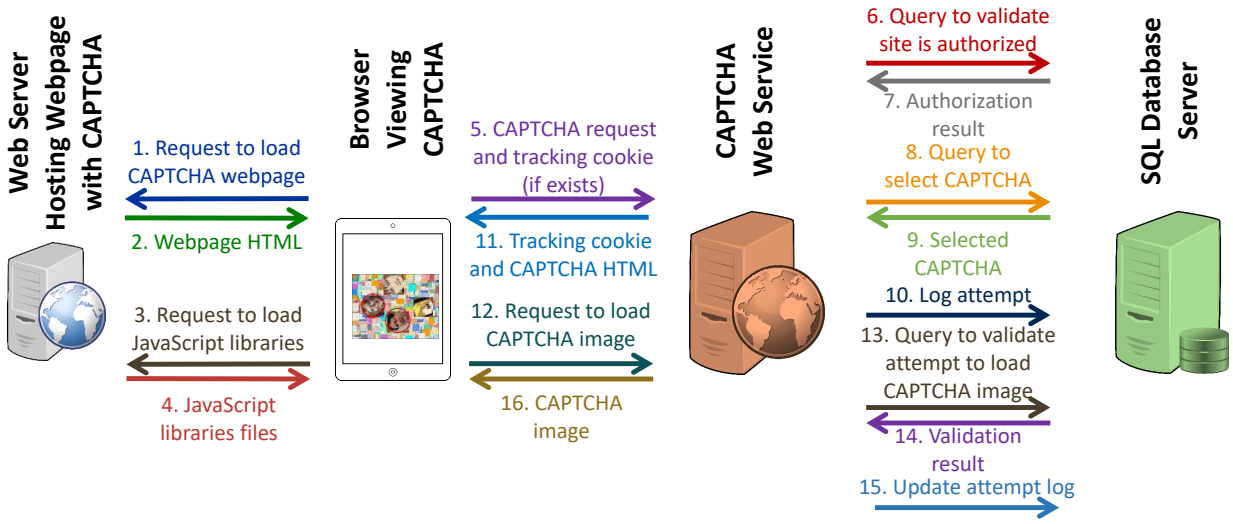


Figure 9.1: Communications between a browser and other servers required to display a CAPTCHA.

The process of displaying a CAPTCHA involves communication between a user's browser and several different servers. Fig. 9.1 illustrates each step in the process.

```
<script src="jquery.min.js"></script>
<script src="jcanvas.min.js"></script>
<script src="bowser.min.js"></script>
<script src="CaptchaApi.es5.min.js"></script>

<form method="post">
<div id="captchaapi_captcha" data-captchaapi_siteguid="site-identifier"
   data-captchaapi_displaygroupname="aicaptcha"></div>
<input type="submit" />
</form>
```

Figure 9.2: HTML code to embed a CAPTCHA

To use a CAPTCHA, a web developer must embed five lines of HTML code into their webpage, as shown in Fig. 9.2. Four lines are `SCRIPT` tags to load the JavaScript browser library and its dependencies to allow the CAPTCHA to be retrieved and to record the user attempt. The fifth line, the CAPTCHA `DIV` tag, configures the CAPTCHA and in-

dicates where it is to be displayed in the webpage. Two data parameters must be specified: (a) `data-captchaapi_siteguid`, a unique identifier to authorize the website to display CAPTCHAs and (b) `data-captchaapi_displaygroupname`, which indicates the type of CAPTCHA to be displayed. The CAPTCHA `DIV` tag must be included inside a form to facilitate submission of attempt data. In most common use cases, such as when the CAPTCHA is used to augment a login page, the CAPTCHA can be added to an existing form.

When users visit a webpage, their browser communicates with the web server hosting the page to download its contents. As the browser loads the document, it identifies additional items such as images and JavaScript code that must also be loaded. The browser then retrieves these items; in the case of a webpage containing a CAPTCHA, this includes the JavaScript browser library and its dependencies.

The browser library automatically runs once the webpage is fully loaded. It parses the document to locate the CAPTCHA `DIV` tag. The data parameters, along with the value of the tracking cookie, if already set, are sent to the CAPTCHA web service to request a new CAPTCHA.

When the web service receives the request for a new CAPTCHA, it first verifies the provided site identifier corresponds with the configured website address and is authorized to display the CAPTCHA. If the website is authorized, the web service queries its database to select a new CAPTCHA of the type requested. If a tracking cookie was provided in the CAPTCHA request, it is used to ensure that a previously selected CAPTCHA is not chosen again; if no tracking cookie was provided, a new one is generated for future use. By eliminating repetitive display of the same CAPTCHAs through use of the tracking cookie, the platform is able to improve its resilience against automated attacks while also allowing for faster evaluation of large CAPTCHA image sets. The selected CAPTCHA is logged in the database along with the tracking cookie and the user's IP address. HTML code is

generated to display the CAPTCHA images and instructions. This HTML code, along with the tracking cookie, is then returned to the browser.

The browser embeds the HTML code it receives from the web service inside of the CAPTCHA `DIV` tag. It displays the updated webpage and makes another request to the web service to load the image for the CAPTCHA test. The web service verifies the request comes from the same IP address as the original CAPTCHA request and that the image has not already been displayed; these checks help to ensure that the CAPTCHA is not diverted to be solved by a third party. If the checks do not indicate any issues, the CAPTCHA image is sent to the browser and the attempt's record is updated to reflect the image has been shown.

When the browser receives the CAPTCHA image, it displays it for the user. The CAPTCHA is now ready to be solved.

Solving CAPTCHAs

(a) Unsolved CAPTCHA

(b) Solved CAPTCHA with blue dots on selected locations

Figure 9.3: Examples of aiCAPTCHA.

A CAPTCHA can be solved once its instructions and image have been displayed. Fig. 9.3a shows how a CAPTCHA appears in its initial unsolved state.

Users complete the CAPTCHA test by clicking (with a mouse) or tapping (on a touch-screen) on the locations in the CAPTCHA image specified in the instructions. Every time the user selects a location within the CAPTCHA image, jCanvas updates the CAPTCHA image to show a blue dot on the selected locations, as depicted in Fig. 9.3b. The browser library also records the coordinates of the selected location, the current time, and other details in a hidden form field created by the web service-provided CAPTCHA HTML code.

When users have finished solving the CAPTCHA, they click a button to submit the form containing the CAPTCHA. Their results are sent to the server hosting the CAPTCHA-containing webpage to determine if the user should be granted access to the resource protected by the CAPTCHA.

Validating CAPTCHA results

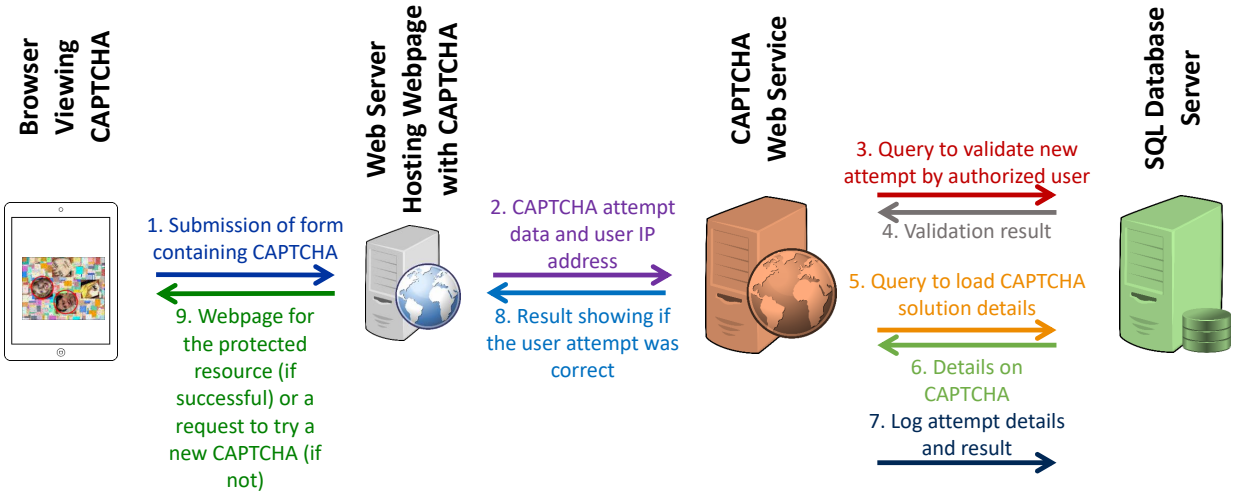


Figure 9.4: Communications between a browser and other servers required to validate a CAPTCHA attempt.

When the web server hosting the CAPTCHA-containing webpage receives a form submission containing a recorded CAPTCHA attempt, it extracts the CAPTCHA information. These recorded details, along with the user's IP address, are sent to the CAPTCHA web service for validation.

The web service first queries its database to ensure the attempt has not been previously submitted and that the submission was from the same IP address that originally requested the CAPTCHA. If both items are valid, indicating the CAPTCHA has not been repeatedly attempted or diverted to a third party, the web service retrieves the solution for the attempted CAPTCHA from the database. Using an algorithm appropriate for the specific CAPTCHA, the user's attempt is verified to determine if it correctly solved the test. Details on the user's submitted attempt are recorded in the database along with the result of whether or not the CAPTCHA was correctly solved.

After details are logged, the CAPTCHA web service sends a message to the hosting web server indicating whether the CAPTCHA was correctly solved. The hosting web server processes this result and takes action accordingly. In general, if the user correctly solved the CAPTCHA, they will be granted access to the resource being protected by the CAPTCHA. If the user's attempt was incorrect, they may be prompted to try solving a new CAPTCHA.

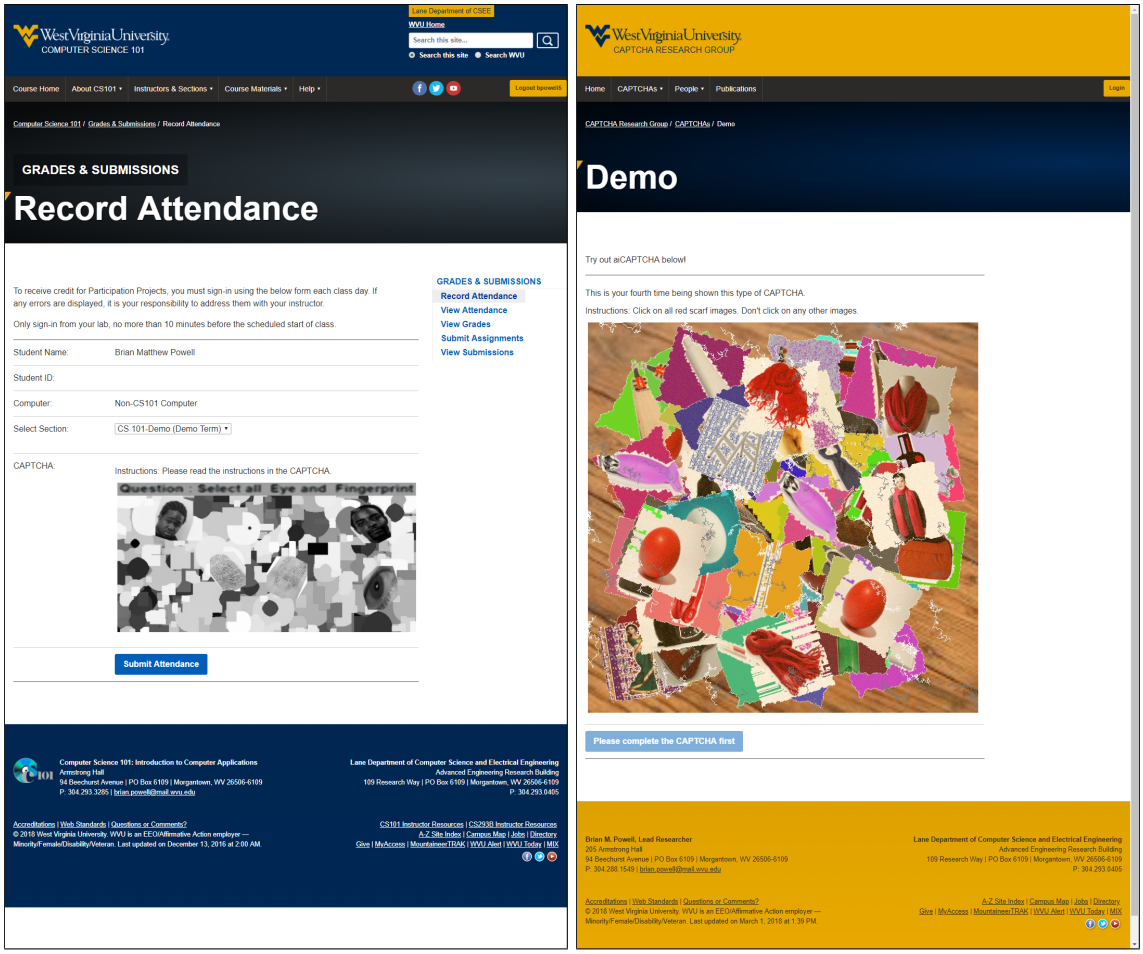
Results analysis

Analysis of collected CAPTCHA data is performed by connecting to the CAPTCHA database, either directly through SQL Server Management Studio [203] or via ODBC-compliant software such as Microsoft Access. These programs allow access to the pre-established database views providing aggregated statistics on recorded CAPTCHA attempts and by several CAPTCHA generation parameters. Custom queries can be created to explore areas of interest in more detail, and the export tools of these programs can facilitate further data analysis with other software.

9.3 Experimental results

The web service at the core of the CAPTCHA Evaluation Platform is currently in use at https://api.captcharesearch.com. This web service, along with the sup-

porting JavaScript browser library and C# hosting server library, are used by two websites to facilitate human performance data collection. On https://cs101.wvu.edu, the CAPTCHA has been integrated into the system students use to record their lecture attendance for the Computer Science 101 course at West Virginia University. The https://www.captcharesearch.com website provides information on each of the CAPTCHAs developed as part of this research, including access to working demonstrations of each CAPTCHA type. Fig. 9.5 shows how these CAPTCHAs appear on their respective websites.



(a) `cs101.wvu.edu` Attendance System (b) `www.captcharesearch.com` demo page

Figure 9.5: Examples of CAPTCHAs deployed using the CAPTCHA Evaluation Platform.

Through the instances embedded on the `cs101.wvu.edu` and `www.captcharesearch.com` websites, the CAPTCHA Evaluation Platform has been used to record 183,230 attempts on

the CAPTCHAs proposed in this research. These attempts were completed using variety of computing devices including desktop computers, laptops, tablets, and smartphones. All modern browsers were successfully tested including Google Chrome, Mozilla Firefox, Microsoft Internet Explorer, Microsoft Edge, Safari, and Opera. The collected results were subsequently analyzed using the Evaluation Platform's analysis module and appear throughout this dissertation.

Chapter 10

Conclusions and Future Work

10.1 Conclusions

While there are a number of existing CAPTCHA implementations in use to protect online systems from automated attacks, these solutions exhibit significant shortcomings. Few provide for quick, accurate completion by humans while remaining resilient against automated attacks. As the world increasingly moves from accessing the Internet using desktop and laptop computers to using smartphones and tablets, many also suffer from usability challenges. The text-based input methods traditionally favored by CAPTCHAs such as MSN [66] and reCAPTCHA [81] are difficult to use with the auto-correct-assisted touchscreens commonly seen on mobile devices. There is a need for modern CAPTCHAs that balance human ease of use across a wide variety of devices with strong security performance.

This research has proposed a series of CAPTCHAs designed to meet these challenges, as well as a CAPTCHA Evaluation Platform designed to facilitate use of these CAPTCHAs. We began by creating Face Detection CAPTCHA as an initial example of a new style of composite image CAPTCHA based on face detection, a task that humans are good at performing but automated algorithms find challenging. Face Detection CAPTCHA served as a test platform for different types of image distortions, simulated attacks on CAPTCHAs, and

human evaluation processes. While it has a comparatively weak performance, with a 71.8% human success rate and 9.3% automated attack success rate, Face Detection CAPTCHA's development provided beneficial insights into images, distortions, and other generation parameters to use for future CAPTCHAs.

FaceDCAPTCHA builds upon Face Detection CAPTCHA, replacing its fixed distortion levels with settings chosen by a gradient descent-based optimization process that incorporates results from human users and automated adversaries. This optimization significantly enhances the CAPTCHA's performance, with FaceDCAPTCHA showing a human success rate of 86.6% and an automated attack success rate of 0% on its final optimized CAPTCHAs. fgCAPTCHA further improves this optimization process by using a genetic learning algorithm and image quality metrics-based simulations to eliminate the need for human testing in the optimization process. With the genetic learning algorithm's ability to customize each individual CAPTCHA, fgCAPTCHA's human success rate improved to 87.9% while maintaining the same 0% attack success rate.

We next proposed FR-CAPTCHA, which uses face recognition rather than face detection as its test. In this CAPTCHA, users are required to identify a pair of embedded faces belonging to the same individual rather than all of the human faces as in the previous CAPTCHAs. Face recognition is a more difficult task for computers to complete than face detection. The change to a more challenging task permits the use of simpler and less intense distortions in FR-CAPTCHA, which has the effect of boosting human performance to a high 94% success rate without sacrificing reliability against automated attackers.

We continued by creating MB-CAPTCHA, which incorporates several biometric modalities in addition to faces. MB-CAPTCHA requires users to identify embedded fingerprints, eyes, and/or faces categorized by gender or race as specified by instructions included in the CAPTCHA image. This adds additional complexity for automated attackers since they must locate and decipher the instructions, identify embedded face, eye, and fingerprint photographs, and in the case of face photographs, categorize them by gender or race. In testing,

no automated attacker was able to successfully complete this task. The added complexity did negatively impact human results, however, as MB-CAPTCHA's human success rate is 83.6% due in large part to difficulties users experienced in accurately categorizing the embedded faces.

Finally, we developed aiCAPTCHA. This approach relies on object recognition tests, asking users to identify embedded objects chosen from 100 different categories with 172 attributes. This expansive set of source images renders attacks by conventional image classification techniques impractical. aiCAPTCHA is designed to thwart attacks based on new machine learning-based methods by incorporating a negative selection-based artificial immune system to remove CAPTCHAs from its test set that may be defeatable. It also incorporates an adaptive filtering mechanism which optimizes the generated CAPTCHA test set to remove CAPTCHAs that have a poor record of human performance. When taken together, these approaches provide a 94.6% human success rate and a 0% automated attack success rate with a strong expectation the CAPTCHA will remain secure well into the future.

10.2 Future research directions

While this work offers several CAPTCHA implementations providing effective online security while ensuring good user experiences, there are additional areas that can be explored to provide further improvements in CAPTCHA performance. Several potential topics are summarized below:

- Each human being has their own personal set of abilities and preferences. One user may excel at the object recognition tasks in aiCAPTCHA, while another might be more successful at solving fgCAPTCHA face detection-based CAPTCHAs, and a third might be fastest at completing text-based CAPTCHAs. Developing a means of automatically assessing which types of CAPTCHAs are best for an individual user and then selec-

tively presenting those CAPTCHAs could significantly help to reduce the friction that CAPTCHAs can cause in the user experience.

- The ability to synthetically generate images including features designed to defeat automatic recognition algorithms could be game-changing. If it were possible to create and embed images that would be miscategorized by automated attackers, distortions and other features which hinder both humans and attackers could be reduced. This would result in CAPTCHAs that are easier and faster for humans to solve without sacrificing resiliency against automated attacks.

- The development of reliable heuristics for identifying automated attackers' behavior when solving CAPTCHAs could provide an added layer of security. In cases where a heuristic suggests that the user might be a bot rather than a person, modifications could be made to the CAPTCHA process such as displaying higher difficulty tests or requiring multiple CAPTCHAs be completed to make it even less likely that an automated attacker would succeed.

Appendix

CAPTCHA demonstrations

Working demonstrations of Face Detection CAPTCHA, FaceDCAPTCHA, fgCAPTCHA, FR-CAPTCHA, MB-CAPTCHA, and aiCAPTCHA, powered by the CAPTCHA Evaluation Platform, are available at http://www.captcharesearch.com.

Dissemination of research results

1. **B. M. Powell**, A. C. Day, R. Singh, M. Vatsa, and A. Noore, "Image-based face detection CAPTCHA for improved security," *Int. J. Multimedia Intell. and Security*, vol. 1, no. 3, pp. 269–284, 2010.

2. G. Goswami, **B. M. Powell**, M. Vatsa, R. Singh, and A. Noore, "FaceDCAPTCHA: Face Detection based Color Image CAPTCHA," *Future Generation Comput. Syst.*, vol. 31, pp. 59–68, Feb. 2014.

3. **B. M. Powell**, G. Goswami, M. Vatsa, R. Singh, and A. Noore, "fgCAPTCHA: Genetically Optimized Face Image CAPTCHA," *IEEE Access*, vol. 2, pp. 473–484, Apr. 2014.

4. G. Goswami, R. Singh, M. Vatsa, **B. M. Powell**, and A. Noore, "Face Recognition CAPTCHA," in *Proc. IEEE 5th Int. Conf. Biometrics: Theory, Appl. and Syst.*, Washington, D.C., Sep. 2012.

5. G. Goswami, **B. M. Powell**, M. Vatsa, R. Singh, and A. Noore, "FR-CAPTCHA: CAPTCHA Based on Recognizing Human Faces," *PLoS ONE*, vol. 9, no. 4, p. e91708, Apr. 2014.

6. **B. M. Powell**, A. Kumar, J. Thapar, G. Goswami, M. Vatsa, R. Singh, and A. Noore, "A Multibiometrics-based CAPTCHA for Improved Online Security," in *Proc. IEEE 8th Int. Conf. Biometrics: Theory, Appl. and Syst.*, Niagara Falls, New York, Sep. 2016.

7. **B. M. Powell**, E. Kalsy, G. Goswami, M. Vatsa, R. Singh, and A. Noore, "Attack-Resistant aiCAPTCHA using a Negative Selection Artificial Immune System," in *Proc. 38th IEEE Symp. Security and Privacy, 2nd Workshop on Bio-inspired Security, Trust, Assurance, and Resilience*, San Jose, California, May 2017.

Bibliography

[1] "Digital in 2018: Global Digital Report," Jan. 2018. [Online]. Available: https://wearesocial.com/blog/2018/01/global-digital-report-2018

[2] M. Lindner, "Global e-marketer set to grow 25% in 2015," *Internet Retailer*, Jul. 2015. [Online]. Available: https://www.internetretailer.com/2015/07/29/global-e-commerce-set-grow-25-2015

[3] B. Butler, "Gartner: 1/3 of consumer data will be stored in the cloud by '16," Jun. 2012. [Online]. Available: http://www.networkworld.com/article/2189595/cloud-computing/gartner--1-3-of-consumer-data-will-be-stored-in-the-cloud-by--16.html

[4] K. Weins, "Cloud Computing Trends: 2016 State of the Cloud Survey," Feb. 2016. [Online]. Available: http://www.rightscale.com/blog/cloud-industry-insights/cloud-computing-trends-2016-state-cloud-survey

[5] T. Olavsrud, "Top cloud Infrastructure-as-a-Service vendors," *CIO*, Jul. 2015. [Online]. Available: http://www.cio.com/article/2947282/cloud-infrastructure/top-cloud-infrastructure-as-a-service-vendors.html

[6] F. Y. Rashid, "The dirty dozen: 12 cloud security threats," *InfoWorld*, Mar. 2016. [Online]. Available: http://www.infoworld.com/article/3041078/security/the-dirty-dozen-12-cloud-security-threats.html

[7] S. Shirali-Shahreza and A. Movaghar, "A New Anti-Spam Protocol Using CAPTCHA," in *Proc. 2007 IEEE Int. Conf. Networking, Sensing, and Control*, London, England, Apr. 2007, pp. 234–238.

[8] J. Pongsajapan, "CAPTCHA-based spam control for content creation systems," U.S. Patent US7 680 891 B1, Mar., 2010.

[9] S. T. Zargar, J. Joshi, and D. Tipper, "A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks," *Commun. Surveys Tuts.*, vol. 15, no. 4, pp. 2046–2069, 2013.

[10] "WVU directory site will require Login credentials starting Sept. 14," Sep. 2017. [Online]. Available: https://it.wvu.edu/news/2017/09/14/wvu-directory-site-will-require-login-credentials-starting-thursday-sept-14

[11] J. Yan, A. Blackwell, R. Anderson, and A. Grant, "Password Memorability and Security: Empirical Results," *IEEE Security & Privacy*, vol. 2, no. 5, pp. 25–31, 2004.

[12] A. Das, J. Bonneau, M. Caesar, N. Borisov, and X. Wang, "The Tangled Web of Password Reuse," in *Proc. 2014 Network and Distributed System Security Symp.*, San Diego, California, Feb. 2014, pp. 23–26.

[13] FireEye, "Recent Zero-Day Exploits," 2017. [Online]. Available: https://www.fireeye.com/current-threats/recent-zero-day-attacks.html

[14] "2016 Cost of Data Breach Study: Global Study," Ponemon Institute, Tech. Rep., Jun. 2016.

[15] E. Kovacs, "DDoS Attacks Cost $40,000 Per Hour: Incapsula," *Security Week*, Nov. 2014. [Online]. Available: http://www.securityweek.com/ddos-attacks-cost-40000-hour-incapsula

[16] L. von Ahn, "Human Computation," PhD thesis, Carnegie Mellon University, Pittsburgh, Pennsylvania, 2005.

[17] S. Shirali-Shahreza and M. H. Shirali-Shahreza, "Bibliography of works done on CAPTCHA," in *Proc. 3rd Int. Conf. Intelligent System and Knowledge Eng.*, vol. 1, Xiamen, China, Nov. 2008, pp. 205–210.

[18] K. A. Kluever, "Evaluating the Usability and Security of a Video CAPTCHA," M.S. thesis, Rochester Institute of Technology, Rochester, New York, 2008.

[19] L. von Ahn, M. Blum, and J. Langford, "Telling Humans and Computers Apart Automatically," *Commun. of ACM*, vol. 47, no. 2, pp. 56–60, 2004.

[20] H. S. Baird and K. Popat, "Human Interactive Proofs and Document Image Analysis," in *Document Anal. Syst. V.* Berlin, Germany: Springer, 2002, pp. 531–537.

[21] Y. Rui and Z. Liu, "Artifacial: Automated reverse turing test using facial features," *Multimedia Syst.*, vol. 9, no. 6, pp. 493–502, 2004.

[22] E. Bursztein, S. Bethard, C. Fabry, J. C. Mitchell, and D. Jurafsky, "How Good Are Humans at Solving CAPTCHAs? A Large Scale Evaluation," in *Proc. 2010 IEEE Symp. Security and Privacy*, Los Alamitos, California, 2010, pp. 399–413.

[23] O. Starostenko, C. Cruz-Perez, F. Uceda-Ponga, and V. Alarcon-Aquino, "Breaking text-based CAPTCHAs with variable word and character orientation," *Pattern Recognition*, vol. 48, no. 4, pp. 1101–1112, Apr. 2015.

[24] G. Reynaga and S. Chiasson, "The usability of CAPTCHAs on smartphones," in *Proc. 2013 Int. Conf. Security and Cryptography*, Reykjavik, Iceland, Jul. 2013, pp. 1–8.

[25] A. Slater and P. C. Quinn, "Face recognition in the newborn infant," *Infant and Child Development*, vol. 10, no. 1-2, pp. 21–24, Mar. 2001.

[26] R. Budiu, "Login Walls Stop Users in Their Tracks," Mar. 2014. [Online]. Available: https://www.nngroup.com/articles/login-walls/

[27] K. Thomas, D. McCoy, C. Grier, A. Kolcz, and V. Paxson, "Trafficking Fraudulent Accounts: The Role of the Underground Market in Twitter Spam and Abuse," in *Proc. 22nd USENIX Security Symp.*, Washington, D.C., Aug. 2013, pp. 195–210.

[28] B. E. McNair, "Security system providing lockout for invalid access attempts," U.S. Patent US5 559 505 A, Sep., 1996.

[29] B. Grobauer, T. Walloschek, and E. Stocker, "Understanding Cloud Computing Vulnerabilities," *IEEE Security & Privacy*, vol. 9, no. 2, pp. 50–57, Mar. 2011.

[30] A. Al-Qayedi, W. Adi, A. Zahro, and A. Mabrouk, "Combined Web/mobile authentication for secure Web access control," in *Proc. 2004 IEEE Wireless Commun. and Networking Conf.*, vol. 2, Atlanta, Georgia, Mar. 2004, pp. 677–681.

[31] D. L. Shoup and R. O'Farrell, "Multi-factor mobile transaction authentication," U.S. Patent US20 130 282 589 A1, Oct., 2013.

[32] B. Ives, K. R. Walsh, and H. Schneider, "The Domino Effect of Password Reuse," *Commun. of ACM*, vol. 47, no. 4, pp. 75–78, Apr. 2004.

[33] S. Motiee, K. Hawkey, and K. Beznosov, "Do Windows Users Follow the Principle of Least Privilege?: Investigating User Account Control Practices," in *Proc. 6th Symp. Usable Privacy and Security*, New York, New York, 2010, pp. 1:1–1:13.

[34] P. Thurrott, "Windows Vista tip of the week: Make UAC less annoying," May 2008. [Online]. Available: http://winsupersite.com/blog/supersite-blog-39/news2/windows-vista-tip-of-the-week-make-uac-less-annoying-138020

[35] L. von Ahn, M. Blum, N. Hopper, and J. Langford, "CAPTCHA: Using Hard AI Problems for Security," in *Advances in Cryptology - EUROCRYPT 2003*, Warsaw, Poland, May 2003, pp. 294–311.

[36] A. M. Turing, "Computing Machinery and Intelligence," *Mind*, vol. 59, no. 236, pp. 433–460, 1950.

[37] A. L. Coates, H. S. Baird, and R. J. Fateman, "Pessimal print: a reverse Turing test," in *Proc. 6th Int. Conf. Document Anal. and Recognition*, vol. 6, Seattle, Washington, Sep. 2001, pp. 1154–1158.

[38] E. Bursztein, M. Martin, and J. Mitchell, "Text-based CAPTCHA Strengths and Weaknesses," in *Proc. 18th ACM Conf. on Comput. and Commun. Security*, Chicago, Illinois, Oct. 2011, pp. 125–138.

[39] K. Chellapilla and P. Y. Simard, "Using Machine Learning to Break Visual Human Interaction Proofs (HIPs)," in *Advances in Neural Inform. Processing Syst. 17*, L. K. Saul, Y. Weiss, and L. Bottou, Eds. Cambridge, Massachusetts: MIT Press, 2005, pp. 265–272.

[40] R. Gafni and I. Nagar, "CAPTCHA - Security affecting User Experience," *Issues in Informing Sci. and Inform. Technol.*, vol. 13, pp. 63–77, Mar. 2016.

[41] M. Naor, "Verification of a human in the loop or Identification via the Turing Test," Sep. 1996. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.50.6383

[42] A. B. Jeng, C.-C. Tseng, D.-F. Tseng, and J.-C. Wang, "A Study of CAPTCHA and Its Application to User Authentication," in *Proc. 2010 Int. Conf. Computational Collective Intell.*, Kaohsiung, Taiwan, Nov. 2010, pp. 433–440.

[43] J. Elson, J. Douceur, J. Howell, and J. Saul, "Asirra: a CAPTCHA that Exploits Interest-Aligned Manual Image Categorization," in *Proc. 14th ACM Conf. Comput. and Commun. Security*, Alexandria, Virginia, Oct. 2007, pp. 366–374.

[44] J. P. Bigham and A. C. Cavender, "Evaluating existing audio CAPTCHAs and an interface optimized for non-visual use," in *Proc. 27th Int. Conf. Human Factors in Comput. Syst.*, Boston, Massachusetts, 2009, pp. 1829–1838.

[45] M. Mohamed, N. Sachdeva, M. Georgescu, S. Gao, N. Saxena, C. Zhang, P. Kumaraguru, P. C. van Oorschot, and W.-B. Chen, "A Three-way Investigation of a game-CAPTCHA: Automated Attacks, Relay Attacks and Usability," in *Proc. 9th ACM Symp. Inform., Comput., and Commun. Security*, Kyoto, Japan, Jun. 2014, pp. 195–206.

[46] C. J. Hernandez-Castro and A. Ribagorda, "Pitfalls in CAPTCHA design and implementation: The Math CAPTCHA, a case study," *Computers & Security*, vol. 29, no. 1, pp. 141–157, Feb. 2010.

[47] H. Nejati, N.-M. Cheung, R. Sosa, and D. C. I. Koh, "DeepCAPTCHA: An Image CAPTCHA Based on Depth Perception," in *Proc. 5th ACM Multimedia Syst. Conf.*, Singapore, Mar. 2014, pp. 81–90.

[48] "Choosing the type of reCAPTCHA," Jun. 2017. [Online]. Available: https://developers.google.com/recaptcha/docs/versions

[49] M. D. Lillibridge, M. Abadi, K. Bharat, and A. Z. Broder, "Method for selectively restricting access to computer systems," U.S. Patent 6 195 698, Feb., 2001.

[50] G. Moy, N. Jones, C. Harkless, and R. Potter, "Distortion Estimation Techniques in Solving Visual CAPTCHAs," in *Proc. 2004 IEEE Comput. Soc. Conf. Comput. Vision and Pattern Recognition*, vol. 2, Washington, D.C., Jun. 2004, pp. 23–28. ©2004 IEEE.

[51] G. Mori and J. Malik, "Recognizing Objects in Adversarial Clutter: Breaking a Visual CAPTCHA," in *Proc. 2003 IEEE Comput. Soc. Conf. Comput. Vision and Pattern Recognition*, vol. 1, Madison, Wisconsin, Jun. 2003, pp. 134–141.

[52] A. Thayananthan, B. Stenger, P. H. S. Torr, and R. Cipolla, "Shape Context and Chamfer Matching in Cluttered Scenes," in *Proc. 2003 IEEE Comput. Soc. Conf. Comput. Vision and Pattern Recognition*, vol. 1, Madison, Wisconsin, Jun. 2003.

[53] M. T. Nayeem, M. M. R. Akand, N. Sakib, and M. W. U. K. Kabir, "Design of a Human Interaction Proof (HIP) using human cognition in contextual natural conversation," in *Proc. IEEE 13th Int. Conf. Cognitive Informat. and Cognitive Comput.*, London, England, Aug. 2014, pp. 146–154. ©2014 IEEE.

[54] M. Chew and H. S. Baird, "BaffleText: a Human Interactive Proof," in *Proc. Document Recognition and Retrieval X*, Santa Clara, California, Jan. 2003, pp. 305–316.

[55] A. El-Nasan and G. Nagy, "On-line handwriting recognition based on bigram co-occurrences," in *Proc. 16th Int. Conf. Pattern Recognition*, vol. 16, Quebec City, Canada, Aug. 2002, pp. 740–743.

[56] P. Baecher, N. Buscher, M. Fischlin, and B. Milde, "Breaking reCAPTCHA: A Holistic Approach via Shape Recognition," in *Future Challenges in Security and Privacy for Academia and Industry*, J. Camenisch, S. Fischer-Hbner, Y. Murayama, A. Portmann, and C. Rieder, Eds. Berlin, Germany: Springer, 2011, vol. 354, pp. 56–67.

[57] "Securimage PHP Captcha," Mar. 2016. [Online]. Available: https://www.phpcaptcha.org/

[58] J. Yu, X. Ma, and T. Han, "Usability Comparison of Text CAPTCHAs Based on English and Chinese," in *Cross-Cultural Design*, Toronto, Canada, Jul. 2016, pp. 130–138.

[59] J. Fagerbeg, "Cracking captchas with neural networks," Sep. 2015. [Online]. Available: http://codepen.io/birjolaxew/post/cracking-captchas-with-neural-networks

[60] J. Yan and A. S. El Ahmad, "A low-cost attack on a Microsoft captcha," in *Proc. 15th ACM Conf. Comput. and Commun. Security*, Alexandria, Virginia, Oct. 2008, pp. 543–554.

[61] K. Chellapilla, K. Larson, P. Y. Simard, and M. Czerwinski, "Computers beat humans at single character recognition in reading based human interaction proofs (hips)," in *Proc. 2nd Conf. E-mail and Anti-Spam*, Stanford, California, Jul. 2005.

[62] S. M. R. S. Beheshti and P. Liatsis, "CAPTCHA Usability and Performance, How to Measure the Usability Level of Human Interactive Applications Quantitatively and Qualitatively?" in *Proc. 2015 Int. Conf. Developments of E-Systems Eng.*, Dubai, United Arab Emirates, Dec. 2015, pp. 131–136. ©2015 IEEE.

[63] H. Gao, M. Tang, Y. Liu, P. Zhang, and X. Liu, "Research on the Security of Microsoft's Two-Layer Captcha," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 7, pp. 1671–1685, Jul. 2017. ©2017 IEEE.

[64] H. S. Baird, M. A. Moll, and S.-Y. Wang, "ScatterType: a legible but hard-to-segment CAPTCHA," in *Proc. 8th Int. Conf. Document Anal. and Recognition*, vol. 2, Seoul, South Korea, 2005, pp. 935–939. ©2005 IEEE.

[65] N. Roshanbin and J. Miller, "ADAMAS: Interweaving unicode and color to enhance CAPTCHA security," *Future Generation Comput. Syst.*, vol. 55, pp. 289–310, Feb. 2016. Reprinted from *Future Generation Computer Systems*, vol. 55, N. Roshanbin and J. Miller, "ADAMAS: Interweaving unicode and color to enhance CAPTCHA security," pp. 289–310, ©2016, with permission from Elsevier.

[66] K. Chellapilla, K. Larson, P. Y. Simard, and M. Czerwinski, "Building Segmentation Based Human-Friendly Human Interaction Proofs (HIPs)," in *Human Interactive Proofs*. Berlin, Germany: Springer, 2005, pp. 1–26.

[67] A. S. El Ahmad, J. Yan, and L. Marshall, "The robustness of a new CAPTCHA," in *Proc. 3rd Eur. Workshop System Security*, Paris, France, 2010, pp. 36–41.

[68] H. S. Baird, M. A. Moll, and S.-Y. Wang, "A Highly Legible CAPTCHA That Resists Segmentation Attacks," in *Human Interactive Proofs*, 2005, pp. 27–41.

[69] A. S. El Ahmad, J. Yan, and W.-Y. Ng, "CAPTCHA Design: Color, Usability, and Security," *IEEE Internet Comput.*, vol. 16, no. 2, pp. 44–51, Mar. 2012.

[70] "BotDetect CAPTCHA Image Samples," 2017. [Online]. Available: https://captcha.com/captcha-examples.html

[71] D. George, W. Lehrach, K. Kansky, M. Lzaro-Gredilla, C. Laan, B. Marthi, X. Lou, Z. Meng, Y. Liu, H. Wang, A. Lavin, and D. S. Phoenix, "A generative vision model that trains with high data efficiency and breaks text-based CAPTCHAs," *Science*, vol. 358, no. 6368, p. eaag2612, Dec. 2017.

[72] C.-C. Hsieh and Z.-Y. Wu, "Anti-SIFT Images Based CAPTCHA Using Versatile Characters," in *2013 Int. Conf. Inform. Sci. and Appl.*, Pattaya, Thailand, Jun. 2013, pp. 1–4.

[73] M. Imsamai and S. Phimoltares, "3d CAPTCHA: A Next Generation of the CAPTCHA," in *Proc. 2010 Int. Conf. Inform. Sci. and Appl.*, Seoul, South Korea, Apr. 2010, pp. 1–8.

[74] "tEABAG_3d," May 2013. [Online]. Available: https://web.archive.org/web/20130518111533/http://www.ocr-research.org.ua/teabag.html

[75] Q. Ye, Y. Chen, and B. Zhu, "The Robustness of a New 3d CAPTCHA," in *Proc. 11th IAPR Int. Workshop Document Anal. Syst.*, Tours-Loire Valley, France, Apr. 2014, pp. 319–323.

[76] V. D. Nguyen, Y.-W. Chow, and W. Susilo, "On the security of text-based 3d CAPTCHAs," *Computers & Security*, vol. 45, pp. 84–99, Sep. 2014.

[77] Y.-W. Chow and W. Susilo, "Enhanced STE3d-CAP: A Novel 3d CAPTCHA Family," in *Information Security Practice and Experience 2012*, vol. 7232, Hangzhou, China, Apr. 2012, pp. 170–181.

[78] A. Rusu and V. Govindaraju, "Handwritten CAPTCHA: using the difference in the abilities of humans and machines in reading handwritten words," in *Proc. 9th Int. Workshop Frontiers in Handwriting Recognition*, Tokyo, Japan, Oct. 2004, pp. 226–231.

[79] A. O. Thomas, A. Rusu, and V. Govindaraju, "Synthetic handwritten CAPTCHAs," *Pattern Recognition*, vol. 42, no. 12, pp. 3365–3373, 2009.

[80] M. H. Aldosari and A. A. Al-Daraiseh, "Strong multilingual CAPTCHA based on handwritten characters," in *Proc. 7th Int. Conf. Inform. and Commun. Syst.*, Irbid, Jordan, Apr. 2016, pp. 239–245.

[81] L. von Ahn, B. Maurer, C. McMillen, D. Abraham, and M. Blum, "reCAPTCHA: Human-Based Character Recognition via Web Security Measures," *Science*, vol. 321, no. 5895, pp. 1465–1468, Sep. 2008.

[82] S. Perez, "Google Now Using ReCAPTCHA To Decode Street View Addresses," *TechCrunch*, Mar. 2012. [Online]. Available: http://social.techcrunch.com/2012/03/29/google-now-using-recaptcha-to-decode-street-view-addresses/

[83] "Captcha Alternatives and thoughts," Dec. 2015. [Online]. Available: https://www.w3.org/WAI/GL/wiki/Captcha_Alternatives_and_thoughts

[84] S. Quarton, "Google's Anti-Spam Strategy: No CAPTCHA reCAPTCHA," *Torque*, Dec. 2014. [Online]. Available: https://torquemag.io/2014/12/googles-anti-spam-strategy-no-captcha-recaptcha/

[85] "Customizing the Look and Feel of reCAPTCHA," May 2016. [Online]. Available: https://developers.google.com/recaptcha/old/docs/customization. This dissertation incorporates content from "Customizing the Look and Feel of recAPTCHA" licensed under the CC BY 3.0 license. The source figure illustrating the red default theme appears unmodified in this dissertation as Fig. 2.3.

[86] E. R. Vimina and A. U. Areekal, "Telling computers and humans apart automatically using activity recognition," in *Proc. 2009 IEEE Int. Conf. Syst., Man and Cybernetics*, San Antonio, Texas, Oct. 2009, pp. 4906–4909. ©2009 IEEE.

[87] M. Chew and J. Tygar, "Image Recognition CAPTCHAs," in *ICS 2004: Information Security*, Palo Alto, California, Sep. 2004, pp. 268–279.

[88] O. Warner, "The Cutest Human-Test: KittenAuth," Apr. 2006. [Online]. Available: http://www.thepcspy.com/read/the_cutest_humantest_kittenauth

[89] P. Lupkowski, "Human computation — how people solve difficult AI problems (having fun doing it)," *Homo Ludens*, vol. 1, no. 3, pp. 81–94, 2011.

[90] H. Arif, H. Hajjdiab, and A. Khalil, "Simple visual CAPTCHA approach," in *Proc. 2016 IEEE Int. Conf. Knowledge Eng. and Appl.*, Singapore, Sep. 2016, pp. 108–112.

[91] "Confident CAPTCHA," 2017. [Online]. Available: http://confidenttechnologies.com/confident-captcha/

[92] S. Wei, Q. Wu, and M. Ren, "Relacha: Using Associative Meaning for Image Captcha Understandability," in *SpaCCS 2017: Security, Privacy, and Anonymity in Computation, Commun., and Storage*, Guangzhou, China, Dec. 2017, pp. 353–367.

[93] P. Golle, "Machine Learning Attacks Against the Asirra CAPTCHA," in *Proc. 15th ACM Conf. Comput. and Commun. Security*, New York, New York, Oct. 2008, pp. 535–542.

[94] D. Lorenzi, J. Vaidya, A. Aich, S. Sural, V. Atluri, and J. Calca, "EmojiTCHA: Using Emotion Recognition to Tell Computers and Humans Apart," in *ICT Syst. Security and Privacy Protection*, Rome, Italy, May 2017, pp. 281–295.

[95] C. Obimbo, A. Halligan, and P. De Freitas, "CaptchAll: An Improvement on the Modern Text-based CAPTCHA," *Procedia Comput. Science*, vol. 20, pp. 496–501, 2013.

[96] P. Matthews and C. C. Zou, "Scene tagging: image-based CAPTCHA using image composition and object relationships," in *Proc. 5th ACM Symp. Inform., Comput. and Commun. Security*, Beijing, China, 2010, pp. 345–350.

[97] R. Datta, J. Li, and J. Z. Wang, "Exploiting the Human-Machine Gap in Image Recognition for Designing CAPTCHAs," *IEEE Trans. Inf. Forensics Security*, vol. 4, no. 3, pp. 504–518, 2009. ©2009 IEEE.

[98] B. B. Zhu, J. Yan, Q. Li, C. Yang, J. Liu, N. Xu, M. Yi, and K. Cai, "Attacks and design of image recognition CAPTCHAs," in *Proc. 17th ACM Conf. Comput. and Commun. Security*, Chicago, Illinois, Oct. 2010, pp. 187–200.

[99] H. S. Baird and J. L. Bentley, "Implicit CAPTCHAs," in *Proc. Document Recognition and Retrieval XII*, vol. 5676, San Jose, California, Jan. 2005, pp. 191–196.

[100] P. Parhi, A. K. Karlson, and B. B. Bederson, "Target Size Study for One-handed Thumb Use on Small Touchscreen Devices," in *Proc. 8th Conf. Human-Comput. Interaction with Mobile Devices and Services*, Espoo, Finland, Sep. 2006, pp. 203–210.

[101] D. Misra and K. Gaj, "Face Recognition CAPTCHAs," in *Proc. Advanced Int. Conf. Telecommun. and Int. Conf. Internet and Web Appl. and Services*, Guadeloupe, French Caribbean, Feb. 2006, p. 122. ©2006 IEEE.

[102] Y. Rui, Z. Liu, S. Kallin, G. Janke, and C. Paya, "Characters or Face: A User Study on Ease of Use for HIPs," in *Proc. 2nd Int. Workshop Human Interactive Proofs*, Bethlehem, Pennsylvania, May 2005, pp. 53–65.

[103] A. Sano, M. Fujita, and M. Nishigaki, "Directcha: A proposal of spatiometric mental rotation CAPTCHA," in *Proc. 14th Annu. Conf. Privacy, Security and Trust*, Auckland, New Zealand, Dec. 2016, pp. 585–592.

[104] K. A. Kluever and R. Zanibbi, "Balancing Usability and Security in a Video CAPTCHA," in *Proc. 5th Symp. Usable Privacy and Security*, Mountain View, California, Jul. 2009, pp. 1–11.

[105] K. Rao, K. Sri, and G. Sai, "A Novel Video CAPTCHA Technique To Prevent BOT Attacks," in *Procedia Comput. Science: Proc. Int. Conf. Computational Modeling and Security 2016*, vol. 85, Bengaluru, India, Feb. 2016, pp. 236–240. [Online]. Available: https://doi.org/10.1016/j.procs.2016.05.220. This dissertation incorporates content from "A Novel Video CAPTCHA Technique To Prevent Bot Attacks" licensed under the CC BY-NC-ND 4.0 license. Source Figs. 4 and 5 appear unmodified in this dissertation as Figs. 2.7a and 2.7b, respectively.

[106] J. Cui, L. Wang, J. Mei, D. Zhang, X. Wang, Y. Peng, and W. Zhang, "CAPTCHA design based on moving object recognition problem," in *Proc. 3rd Int. Conf. Inform. Sciences and Interaction Sciences*, Chengdu, China, Jun. 2010, pp. 158–162.

[107] J. Cui, W. Zhang, Y. Peng, Y. Liang, B. Xiao, J. Mei, D. Zhang, and X. Wang, "A 3-layer Dynamic CAPTCHA Implementation," in *Proc. 2nd Int. Workshop Edu. Technol. and Comput. Sci.*, vol. 1, Wuhan, China, Mar. 2010, pp. 23–26.

[108] J. Kirk, "NuCaptcha Improves Integration of CAPTCHA System," *CIO*, Aug. 2011. [Online]. Available: http://www.cio.com/article/2405514/security0/nucaptcha-improves-integration-of-captcha-system.html

[109] S. Gao, M. Mohamed, N. Saxena, and C. Zhang, "Emerging-Image Motion CAPTCHAs: Vulnerabilities of Existing Designs, and Countermeasures," *IEEE Trans. Dependable and Secure Computing*, vol. Preprint, Jun. 2017. [Online]. Available: https://doi.org/10.1109/TDSC.2017.2719031

[110] Y. Xu, G. Reynaga, S. Chiasson, J.-M. Frahm, F. Monrose, and P. van Oorschot, "Security and Usability Challenges of Moving-Object CAPTCHAs: Decoding Codewords in Motion," in *Proc. 21st USENIX Security Symp.*, Bellevue, Washington, Aug. 2012.

[111] "ReBreakCaptcha: Breaking Google's ReCaptcha v2 using... Google," Feb. 2017. [Online]. Available: https://east-ee.com/2017/02/28/rebreakcaptcha-breaking-googles-recaptcha-v2-using-google/

[112] G. Kochanski, D. Lopresti, and C. Shih, "A Reverse Turing Test Using Speech," in *Proc. 7th Int. Conf. Spoken Language Processing*, Denver, Colorado, Sep. 2002, pp. 1357–1360.

[113] T.-Y. Chan, "Using a test-to-speech synthesizer to generate a reverse Turing test," in *Proc. 15th IEEE Int. Conf. on Tools with Artificial Intell.*, Sacramento, California, Nov. 2003, pp. 226–232.

[114] J. Tam, S. Hyde, J. Simsa, and L. von Ahn, "Breaking Audio CAPTCHAs," in *Advances in Neural Inform. Process. Syst.*, vol. 22, Vancouver, Canada, Dec. 2008.

[115] E. Bursztein and S. Bethard, "Decaptcha: Breaking 75% of eBay Audio CAPTCHAs," in *Proc. 3rd USENIX Workshop Offensive Technologies*, Montreal, Canada, Aug. 2009.

[116] S. Ruan, J. O. Wobbrock, K. Liou, A. Ng, and J. Landay, "Speech Is 3x Faster than Typing for English and Mandarin Text Entry on Mobile Devices," *arXiv*, vol. 1608.07323, Aug. 2016.

[117] H. Meutzner, S. Gupta, and D. Kolossa, "Constructing Secure Audio CAPTCHAs by Exploiting Differences Between Humans and Machines," in *Proc. 33rd Annu. ACM Conf. Human Factors in Comput. Syst.*, Seoul, South Korea, Apr. 2015, pp. 2335–2338.

[118] M. Yamaguchi and H. Kikuchi, "Audio-CAPTCHA with distinction between random phoneme sequences and words spoken by multi-speaker," in *Proc. 2017 IEEE Int. Conf. Syst., Man, and Cybernetics*, Banff, Canada, Oct. 2017, pp. 3071–3076.

[119] G. Sauer, J. Holman, J. Lazar, H. Hochheiser, and J. Feng, "Accessible privacy and security: a universally usable human-interaction proof tool," *Universal Access in Inform. Soc.*, vol. 9, no. 3, pp. 239–248, Aug. 2010.

[120] J. H. Ryu, N. Y. Kim, S. Y. Moon, and J. H. Park, "Telling Computer and Human Apart: Image-Sound Based CAPTCHA System," in *MUE 2017/FutureTech 2017: Advanced Multimedia and Ubiquitous Eng.*, Seoul, South Korea, May 2017, pp. 733–736.

[121] J. Lazar, J. Feng, T. Brooks, G. Melamed, B. Wentz, J. Holman, A. Olalere, and N. Ekedebe, "The SoundsRight CAPTCHA: An Improved Approach to Audio Human Interaction Proofs for Blind Users," in *Proc. 2012 SIGCHI Conf. Human Factors in Comput. Syst.*, Austin, Texas, May 2012, pp. 2267–2276.

[122] K. S. Kuppusamy and G. Aghila, "HuMan: an accessible, polymorphic and personalized CAPTCHA interface with preemption feature tailored for persons with visual impairments," *Universal Access in Inform. Soc.*, pp. 1–24, Aug. 2017.

[123] H. Gao, H. Liu, D. Yao, X. Liu, and U. Aickelin, "An Audio CAPTCHA to Distinguish Humans from Computers," in *Proc. 3rd Int. Symp. Electronic Commerce and Security*, Guangzhou, China, Jul. 2010, pp. 265–269.

[124] A. Basso and S. Sicco, "Preventing massive automated access to web resources," *Computers & Security*, vol. 28, no. 3-4, pp. 174–188, May 2009.

[125] N. A. Shah and M. T. Banday, "Drag and Drop Image CAPTCHA," *Sprouts: Working Papers on Inform. Syst.*, vol. 8, no. 46, 2008.

[126] A. A. Galib and R. Safavi-Naini, "MOVTCHA: A CAPTCHA Based on Human Cognitive and Behavioral Features Analysis," in *Inform. and Commun. Security*, Irbid, Jordan, Dec. 2014, pp. 290–304.

[127] B. M. Jakobsson, J. R. Palmer, and G. Maldonado, "Interactive captcha," U.S. Patent US20 160 224 783 A1, Aug., 2016.

[128] W. Zhang, "Zhang's CAPTCHA architecture based on intelligent interaction via RIA," in *Proc. 2nd Int. Conf. Comput. Eng. and Technol.*, vol. 6, Jodhpur, India, Nov. 2010, pp. V6–57–V6–62.

[129] "HTML Drag and Drop API," May 2017. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/HTML_Drag_and_Drop_API

[130] S. A. Ross, J. A. Halderman, and A. Finkelstein, "Sketcha: a captcha based on line drawings of 3d models," in *Proc. 19th Int. Conf. World Wide Web*, Raleigh, North Carolina, 2010, pp. 821–830.

[131] N. R. W. W. M. R. D. Wickramasingha, H. B. R. A. K. R. A. M. Keerawella, S. A. N. Samarasinghe, and R. G. Ragel, "RotateCAPTCHA a novel interactive CAPTCHA design targeting mobile devices," in *Proc. IEEE 10th Int. Conf. Ind. and Inform. Syst.*, Peradeniya, Sri Lanka, Dec. 2015, pp. 49–54.

[132] M. Conti, C. Guarisco, and R. Spolaor, "CAPTCHaStar! A Novel CAPTCHA Based on Interactive Shape Discovery," in *Appl. Cryptography and Network Security*, London, England, Jun. 2016, pp. 611–628.

[133] T.-I. Yang, C.-S. Koong, and C.-C. Tseng, "Game-based image semantic CAPTCHA on handset devices," *Multimedia Tools and Appl.*, vol. 74, no. 14, pp. 5141–5156, Jul. 2015.

[134] "PlayThru: A CAPTCHA Alternative from Are You a Human - developer.force.com," May 2013. [Online]. Available: https://developer.salesforce.com/page/PlayThru:_A_CAPTCHA_Alternative_from_Are_You_a_Human

[135] S. Gao, M. Mohamed, N. Saxena, and C. Zhang, "Gaming the game: Defeating a game captcha with efficient and robust hybrid attacks," in *Proc. 2014 IEEE Int. Conf. Multimedia and Expo*, Chendu, China, Jul. 2014, pp. 1–6.

[136] E. Ogg, "Why flash didn't work out on mobile devices," Nov. 2011. [Online]. Available: https://gigaom.com/2011/11/09/why-flash-didnt-work-out-on-mobile-devices/

[137] T. Yamamoto, J. Tygar, and M. Nishigaki, "CAPTCHA Using Strangeness in Machine Translation," in *Proc. 24th IEEE Int. Conf. Advanced Inform. Networking and Appl.*, Perth, Australia, 2010, pp. 430–437.

[138] C. Pope and K. Kaur, "Is it human or computer? Defending e-commerce with Captchas," *IT Professional*, vol. 7, no. 2, pp. 43–49, 2005. ©2005 IEEE.

[139] M. Osadchy, J. Hernandez-Castro, S. Gibson, O. Dunkelman, and D. Prez-Cabo, "No Bot Expects the DeepCAPTCHA! Introducing Immutable Adversarial Examples, With Applications to CAPTCHA Generation," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 11, pp. 2640–2653, Nov. 2017. ©2017 IEEE.

[140] P. Lupkowski and M. Urbanski, "SemCAPTCHA - user-friendly alternative for OCR-based CAPTCHA systems," in *Proc. Int. Multiconference Comput. Science and Inform. Tech.*, Wisla, Poland, Oct. 2008, pp. 325–329. ©2008 IEEE.

[141] S. Vikram, Y. Fan, and G. Gu, "SEMAGE: A new image-based two-factor CAPTCHA," in *Proc. 27th Annu. Comput. Security Appl. Conf.*, Orlando, Florida, Dec. 2011, p. 237.

[142] S. Sivakorn, J. Polakis, and A. D. Keromytis, "I'm not a human: Breaking the Google reCAPTCHA," in *Black Hat Asia 2016*, Singapore, Mar. 2016.

[143] S. Sivakorn, I. Polakis, and A. D. Keromytis, "I am Robot: (Deep) Learning to Break Semantic Image CAPTCHAs," in *Proc. 2016 IEEE Eur. Symp. Security and Privacy*, Saarbrucken, Germany, Mar. 2016, pp. 388–403. ©2016 IEEE.

[144] V. Shet, "Are you a robot? Introducing No CAPTCHA reCAPTCHA," Dec. 2014. [Online]. Available: https://security.googleblog.com/2014/12/are-you-robot-introducing-no-captcha.html

[145] R. Whitwam, "Google's new bot-stopping reCAPTCHA is completely invisible," *ExtremeTech*, Mar. 2017. [Online]. Available: https://www.extremetech.com/internet/245685-googles-new-bot-stopping-recaptcha-completely-invisible

[146] R. Verger, "Google just made the internet a tiny bit less annoying," *Popular Science*, Mar. 2017. [Online]. Available: http://www.popsci.com/google-invisible-recaptcha

[147] D. Uria, "Googly-eyed robot beats 'I am not a robot' Captcha test," *UPI*, Jan. 2017. [Online]. Available: https://www.upi.com/Googly-eyed-robot-beats-I-am-not-a-robot-Captcha-test/2861485544129/

[148] B. M. Powell, A. C. Day, R. Singh, M. Vatsa, and A. Noore, "Image-based face detection CAPTCHA for improved security," *Int. J. Multimedia Intelligence and Security*, vol. 1, no. 3, pp. 269–284, Inderscience Publishers, 2010.

[149] A. G. Weber, "SIPI Image Database," University of Southern California, Los Angeles, California, Tech. Rep. 315, Apr. 2006.

[150] M. Nixon and A. S. Aguado, *Feature Extraction & Image Processing*, 2nd ed. Academic Press, Jan. 2008.

[151] Carnegie Mellon University, "Carnegie Mellon University Image Data Base: Frontal Face Images," Nov. 2002. [Online]. Available: http://vasc.ri.cmu.edu//idb/html/face/frontal_images/index.html

[152] Flickr, "Flickr," 2010. [Online]. Available: http://www.flickr.com/

[153] P. Viola and M. Jones, "Robust real-time object detection," *Int. J. Comput. Vision*, vol. 57, no. 2, pp. 137–154, 2002.

[154] C. Zhang and Z. Zhang, "Boosting-Based Face Detection and Adaptation," *Synthesis Lectures Comput. Vision*, vol. 2, pp. 1–140, Sep. 2010.

[155] Z. Wang and A. Bovik, "A Universal Image Quality Index," *IEEE Signal Process. Lett.*, vol. 9, no. 3, pp. 81–84, Mar. 2002.

[156] H. Sheikh and A. Bovik, "Image Information And Visual Quality," *IEEE Trans. Image Process.*, vol. 15, no. 2, pp. 430–444, 2006.

[157] G. Goswami, B. M. Powell, M. Vatsa, R. Singh, and A. Noore, "FaceDCAPTCHA: Face Detection based Color Image CAPTCHA," *Future Generation Comput. Syst.*, vol. 31, pp. 59–68, Feb. 2014.

[158] D. Marsh, *Applied Geometry for Computer Graphics and CAD*, 2nd ed. Berlin, Germany: Springer, 2005.

[159] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 2nd ed. Upper Saddle River, New Jersey: Prentice Hall, 2002.

[160] "Picasa," 2014. [Online]. Available: http://picasa.google.com

[161] G. B. Huang, M. Mattar, T. Berg, and E. Learned-Miller, "Labeled faces in the wild: A database forstudying face recognition in unconstrained environments," in *Proc. Workshop Faces in "Real-Life" Images: Detection, Alignment, and Recognition*, Marseille, France, 2008.

[162] "Photobucket," 2014. [Online]. Available: http://photobucket.com

[163] B. M. Powell, G. Goswami, M. Vatsa, R. Singh, and A. Noore, "fgCAPTCHA: Genetically Optimized Face Image CAPTCHA," *IEEE Access*, vol. 2, pp. 473–484, Apr. 2014. ©2014 IEEE.

[164] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image Quality Assessment: From Error Visibility to Structural Similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

[165] A. McAndrew, *An Introduction to Digital Image Processing with Matlab*. Boston, Massachusetts: Course Technology, 2004.

[166] M. Mitchell, *An Introduction to Genetic Algorithms*. Cambridge, Massachusetts: The MIT Press, 1998.

[167] S. Sivanandam and S. Deepa, *Introduction to Genetic Algorithms*. Berlin, Germany: Springer, 2008.

[168] G. Goswami, B. M. Powell, R. Singh, M. Vatsa, and A. Noore, "Face Recognition CAPTCHA," in *Proc. IEEE 5th Int. Conf. Biometrics: Theory, Appl. and Syst.*, Washington, D.C., Sep. 2012. ©2012 IEEE.

[169] G. Goswami, B. M. Powell, M. Vatsa, R. Singh, and A. Noore, "FR-CAPTCHA: CAPTCHA Based on Recognizing Human Faces," *PLoS ONE*, vol. 9, no. 4, p. e91708, Apr. 2014. [Online]. Available: https://doi.org/10.1371/journal.pone.0091708. This dissertation incorporates content from "FR-CAPTCHA: CAPTCHA Based on Recognizing Human Faces" licensed under the CC BY 4.0 license. Source Figs. 1 and 2 appear unmodified in this dissertation as Figs. 6.1 and 6.3, respectively. A portion of Source Fig. 3 appears in this dissertation as Fig. 6.4. Source Fig. 5 has been rearranged to appear in this disseration as Fig. 6.5. Source Tables 1 and 2 appear visually reformatted in this dissertation as Tables 6.1 and 6.2.

[170] P. Sinha, B. Balas, Y. Ostrovsky, and R. Russell, "Face Recognition by Humans: Nineteen Results All Computer Vision Researchers Should Know About," *Proc. IEEE*, vol. 94, no. 11, pp. 1948–1962, Nov. 2006.

[171] J. Ruiz-del Solar, R. Verschae, and M. Correa, "Recognition of Faces in Unconstrained Environments: A Comparative Study," *EURASIP J. Adv. Signal Process.*, vol. 2009, no. 1, p. 184617, Dec. 2009.

[172] R. A. Johnston and A. J. Edmonds, "Familiar and unfamiliar face recognition: A review," *Memory*, vol. 17, no. 5, pp. 577–596, Jul. 2009.

[173] A. R. Martinez and R. Benavente, "The AR Face Database," Computer Vision Center, Technical Report 24, Jun. 1998.

[174] R. Datta, J. Li, and J. Z. Wang, "IMAGINATION: a robust image-based CAPTCHA generation system," in *Proc. 13th Annu. ACM Int. Conf. Multimedia*, Singapore, Nov. 2005, pp. 331–334.

[175] B. M. Powell, A. Kumar, J. Thapar, G. Goswami, M. Vatsa, R. Singh, and A. Noore, "A Multibiometrics-based CAPTCHA for Improved Online Security," in *Proc. IEEE*

8th Int. Conf. on Biometrics: Theory, Appl. and Syst., Niagara Falls, New York, Sep. 2016. ©2016 IEEE.

[176] E. R. Dougherty and R. A. Lotufo, *Hands-On Morphological Image Processing*. Bellingham, Washington: SPIE Press, 2003.

[177] I. Megvii, "Face++: Leading Face Recognition on Cloud," 2015. [Online]. Available: http://www.faceplusplus.com/

[178] R. Vazan, "SourceAFIS," 2015. [Online]. Available: http://www.sourceafis.org/

[179] K. Ricanek and T. Tesafaye, "MORPH: a longitudinal image database of normal adult age-progression," in *Proc. 7th Int. Conf. Automat. Face and Gesture Recognition*, Southampton, England, Apr. 2006, pp. 341–345.

[180] A. Sharma, S. Verma, M. Vatsa, and R. Singh, "On Cross Spectral Periocular Recognition," in *Proc. 2014 IEEE Int. Conf. Image Processing*, Paris, France, Oct. 2014.

[181] D. Maltoni, M. Maio, A. K. Jain, and S. Prabhakar, *Handbook of Fingerprint Recognition*, 2nd ed. London, England: Springer, 2009.

[182] B. M. Powell, E. Kalsy, G. Goswami, M. Vatsa, R. Singh, and A. Noore, "Attack-Resistant aiCAPTCHA using a Negative Selection Articial Immune System," in *Proc. 38th IEEE Symp. on Security and Privacy, 2nd Workshop on Bio-inspired Security, Trust, Assurance, and Resilience*, San Jose, California, May 2017. ©2017 IEEE.

[183] Y. P. Chen, C. H. Liu, K. Y. Chou, and S. Y. Wang, "Real-time and low-memory multi-face detection system design based on naive Bayes classifier using FPGA," in *Proc. 2016 Int. Automat. Control Conf.*, Taichung, Taiwan, Nov. 2016, pp. 7–12.

[184] D. Triantafyllidou and A. Tefas, "Face detection based on deep convolutional neural networks exploiting incremental facial part learning," in *Proc. 23rd Int. Conf. Pattern Recognition*, Dec. 2016, pp. 3560–3565.

[185] V. Nair and G. E. Hinton, "3d Object Recognition with Deep Belief Nets," in *Advances in Neural Inform. Processing Syst. 22*, Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, Eds. Red Hook, New York: Curran Associates, 2009, pp. 1339–1347.

[186] T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, and T. Poggio, "Robust Object Recognition with Cortex-Like Mechanisms," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 3, pp. 411–426, Mar. 2007.

[187] J. Winn, A. Criminisi, and T. Minka, "Object categorization by learned universal visual dictionary," in *Proc. 10th IEEE Int. Conf. Comput. Vision*, vol. 2, Beijing, China, Oct. 2005, pp. 1800–1807.

[188] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," in *Proc. 2005 IEEE Comput. Soc. Conf. Comput. Vision and Pattern Recognition*, San Diego, California, Jun. 2005, pp. 886–893.

[189] J. Brownlee, *Clever Algorithms: Nature-Inspired Programming Recipes*. Raleigh, North Carolina: Lulu.com, 2012.

[190] S. Forrest, A. S. Perelson, L. Allen, and R. Cherukuri, "Self-Nonself Discrimination in a Computer," in *Proc. 1994 IEEE Symp. Security and Privacy*, Oakland, California, May 1994, pp. 202–212.

[191] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[192] ——, "Object recognition from local scale-invariant features," in *Proc. 7th IEEE Int. Conf. Comput. Vision*, vol. 2, Kerkyra, Greece, Sep. 1999, pp. 1150–1157.

[193] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual categorization with bags of keypoints," in *Proc. 8th Eur. Conf. Comput. Vision, Workshop Statistical Learning in Comput. Vision*, vol. 1, Prague, Czech Republic, May 2004, pp. 1–2.

[194] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object Detection with Discriminatively Trained Part-Based Models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 9, pp. 1627–1645, Sep. 2010.

[195] R. B. Girshick, P. F. Felzenszwalb, and D. McAllester, "Discriminatively trained deformable part models, release 5," Sep. 2012. [Online]. Available: http://people.cs.uchicago.edu/~rbg/latent-release5/

[196] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv*, vol. 1409.1556, Sep. 2014.

[197] "Humans + porn = solved Captcha," *Network Security*, vol. 2007, no. 11, p. 2, Nov. 2007.

[198] A. Rodriguez, "Restful web services: The basics," *IBM developerWorks*, pp. 1–11, Nov. 2008. [Online]. Available: http://www.gregbulla.com/TechStuff/Docs/ws-restful-pdf.pdf

[199] D. Roth, R. Anderson, and S. Luttin, "Introduction to ASP.NET Core," Dec. 2017. [Online]. Available: https://docs.microsoft.com/en-us/aspnet/core/

[200] "jQuery," 2018. [Online]. Available: https://jquery.com/

[201] C. Evans, "jCanvas," 2018. [Online]. Available: https://projects.calebevans.me/jcanvas/jcanvas/

[202] D. Demchenko, "bowser: a browser detector," Feb. 2018. [Online]. Available: https://github.com/lancedikson/bowser

[203] "SQL Server Management Studio (SSMS)," Feb. 2018. [Online]. Available: https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms